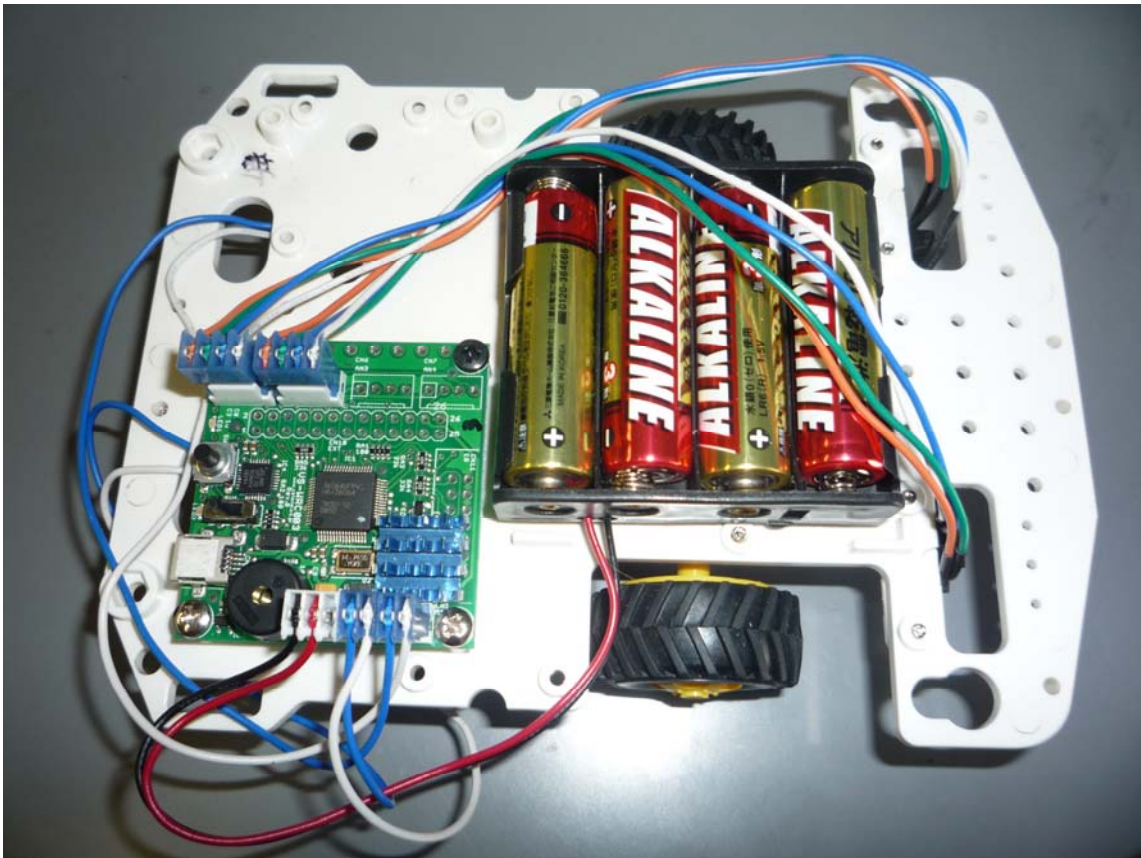
教育用ロボット

「BeautoChaser」

でC言語を学ぼう

平成22年度卒業
山口大学教育学部情報科学教育課程表現情報処理コース情報工学研究室
石原　知枝

目次

1. ロボットを動かすための環境を設定しよう

1.1. HEW のインストールとセットアップ

1.1.1. ルネサスのユーザ登録

手順①　ルネサス Web ページ内のマイルネサスページで新規登録をクリックすると，個人情報の入力画面になります．

図 1-1　新規登録画面へ

手順②　必要事項を入力後，ルネサスのプライバシーポリシーを読んだ後，送信ボタンを押します．

図 1-2　必要事項入力画面へ

手順③　ニュースレターの登録画面になりますが，特に興味が無い場合はそのまま送信ボタンを押して次へ行きましょう．

図 1-3　必要事項入力画面へ

手順④　すると「E メールを送付しましたのでご確認下さい」という表示が出るので，個人情報の入力画面で登録したメールアドレスにルネサスからのメールが来ていることを確認してください．

図 1-4　メール確認画面

手順⑤　届いたメールアドレスに書かれた URL をクリックし，表示された Web ページにログイン ID とパスワードを入力したら送信ボタンを押してください. これで登録完了です.

GLOBAL SITE

RENESAS
Everywhere you imagine

登録確認

ご登録されたパスワードをご入力ください。送信ボタンをクリックした後、ログインIDとパスワードが有効になりお客様のプロフィールの変更が可能となります。

図 1-5　ログイン画面

1.1.2. 開発環境の入手

ルネサステクノロジ社のホームページから HEW，フラッシュ開発ツールキット(FDT)を入手します．以下の手順に従って作業を行いましょう．

＜HEW＞

手順①　以下の URL 先を入力してください．

http://japan.renesas.com/fmwk.jsp?cnt=/download_search_results.jsp&fp=/support/downloads/download_results&layerId=2296

手順②　キワード欄に

[無償評価版]　H8SX　H8S　H8　C/C++コンパイラパッケージ

と入力し，検索しましょう．

図 1-6　ダウンロードするソフトウェアを検索する

手順③　まず HEW のダウンロードを行いましょう．

(V.6.02 Renesas 01 以上の一番新しいバージョンをダウンロードしてください)

図 1-7　ダウンロードするソフトウェアを選択する

手順④　ユーザ登録した ID，パスワードを入力してログインします.

ログイン

ー 処理を続ける為にログインしてください。

*の項目は必須入力となっています。

*ログインID:

*パスワード:

送信

●ログインの詳細をお忘れの場合

パスワードをお忘れの場合 ここをクリック

●新規のお客様

新規のお客様はこちらをクリックして登録してください。
（登録の仕方についてはこちらをクリックして確認できます。）

図 1-8　パスワードと ID でのログイン

手順⑤　規約に同意して[Submit]を押します.

6. 弊社は品質・信頼性の向上に努めておりますが、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は、事前に弊社営業担当迄ご相談をお願い致します。

7. 設計に際しては、特に最大定格、動作電源電圧範囲、放熱特性、実装条件及びその他諸条件につきましては、弊社保証範囲内でご使用いただきますようお願い致します。保証値を超えてご使用された場合の故障及び事故につきましては、弊社はその責を負いません。また保証値内のご使用であっても半導体製品について通常予測される故障発生率、故障モードをご考慮の上、弊社製品の動作が原因でご使用機器が人身事故、火災事故、その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。

8. 本製品は耐放射線設計をしておりません。

9. 本サイトに記載された情報の一部または、全部を弊社の文書による承認なしに、転載または複製することを堅くお断り致します。

10. 本サイトにアクセスする際に付与されたユーザーID、パスワードに関しては、第三者に一切開示しないと共に、意図せぬ漏洩を防止する管理を徹底願います。

○ 同意します ○ 同意しません Submit

 Webサイト利用条件 | プライバシーポリシー | RSS | サイトマップ

図 1-9　規約に同意

手順⑥　[Download]を押してインストーラをダウンロードします．

- **動作環境について**

ホストマシン名	OS名	ハードディスク容量
IBM PC/AT ※1 互換機	※2 Windows XP, 2000	240Mバイト以上の空き容量が必要です。

※1. IBM、ATは米国International Business Machines Corporationの登録商標です。
※2. Windowsは米国Microsoft Corporationの登録商標です。

インストール方法

ダウンロードしたファイルを実行してください。インストーラーが起動します(作業を行うディレクトリは、十分な容量のあるドライブをご使用ください)。その後の作業は、インストーラーの指示に従ってください。

無償評価版ソフトウエアご使用にあたって

- 使用権許諾契約書 を必ずお読みください。

ダウンロード

ダウンロード製品名	ファイル名	ファイルサイズ	リンク
【無償評価版】H8SX,H8S,H8ファミリ用C/C++コンパイラパッケージ V.6.02 Release 01	h8v6201_ev.exe	148,820,968 bytes (141.92 Mbytes)	Download

図 1-10　ダウンロードの開始

手順⑦　同様に【無償評価版】フラッシュ開発ツールキット（実行ファイルをマイコンに送るソフトウェア，以下 FDT）も【無償評価版】フラッシュ開発ツールキットと書かれたリンクからダウンロードしてください．

図 1-11　FDT も同様にしてダウンロード

手順⑤　H8SX, H8S, H8 C/C++コンパイラパッケージ，フラッシュ開発ツールキット（FDT）でそれぞれのダウンロードが完了すると，以下の指定した保存先に存在することを確認してください．

○　**H8SX, H8S, H8 C/C++コンパイラパッケージ**

「h8v****_ev.exe」(****にはバージョンの数字が入ります)

○　**フラッシュ開発ツールキット**

「fdtv***r**.exe」(****にはバージョンの数字が入ります)

1.1.3.　HEW のインストール

それでは，開発環境の HEW を以下の手順でインストールしましょう．

手順①　ダウンロードしたインストーラ「h8v****_ev.exe」をダブルクリックし起動します．(****にはバージョンの数字が入ります)

手順②　[Next]を押します．

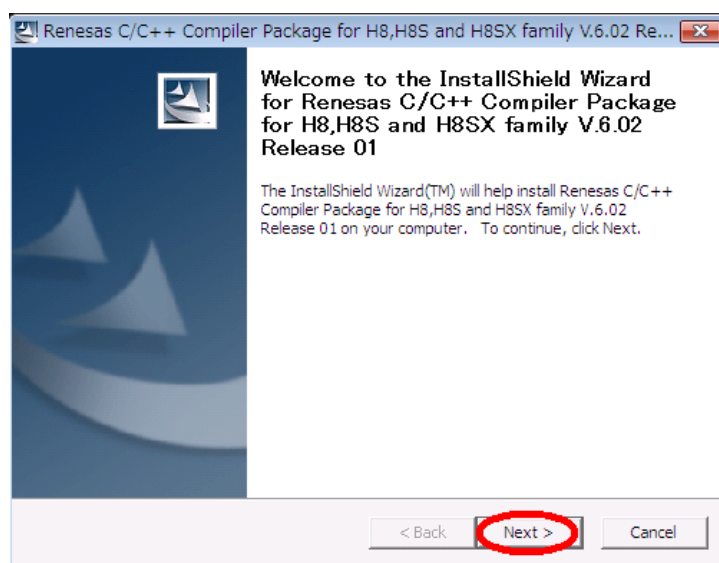



図 1-12　起動画面

手順③　インストールする前に，インストーラのパッケージを展開する必要があります．任意のファイルを指定して[Next]を押してください．

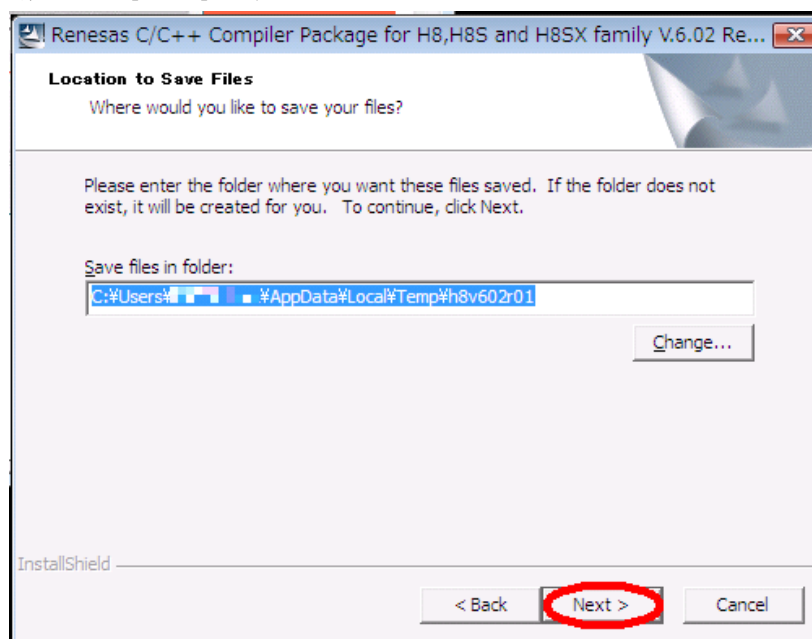



図 1-13　インストーラの展開場所を指定

手順④　インストールの展開が完了すると以下の画面［インストールマネージャ］が表示されます．ここでは，標準インストールを行いますので，［標準インストール］ボタンを押します．

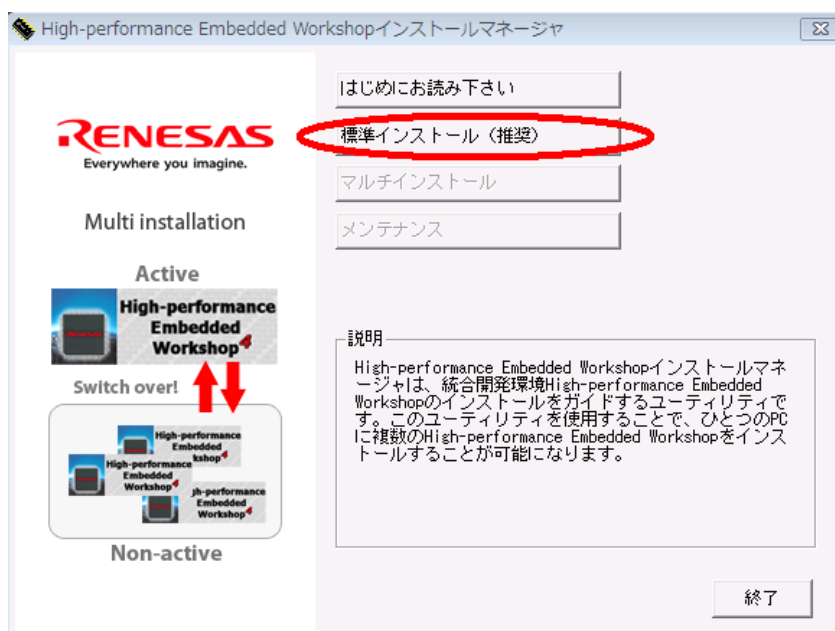



図 1-14　[標準インストール]を指定

手順⑤　以下の画面(インストール製品の選択)にて，チェックボタンを同じように選択し，[インストール］を押します．

オートアップデートを有効にしたい場合は，[オートアップデートユーティリティ］にチェックを入れてください．

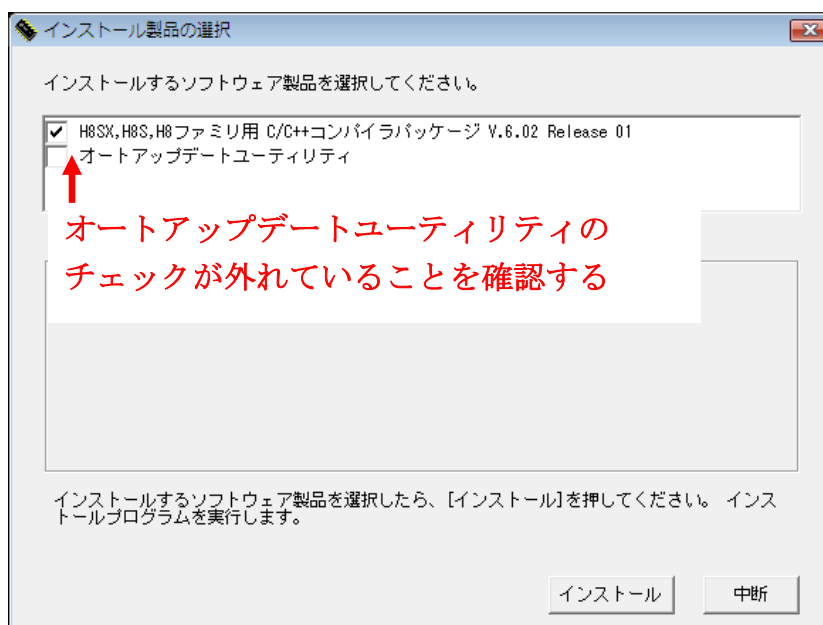



図 1-15 ［オートアップデートユーティリティ］を選択してインストール開始

手順⑥ ［次へ］を押します．

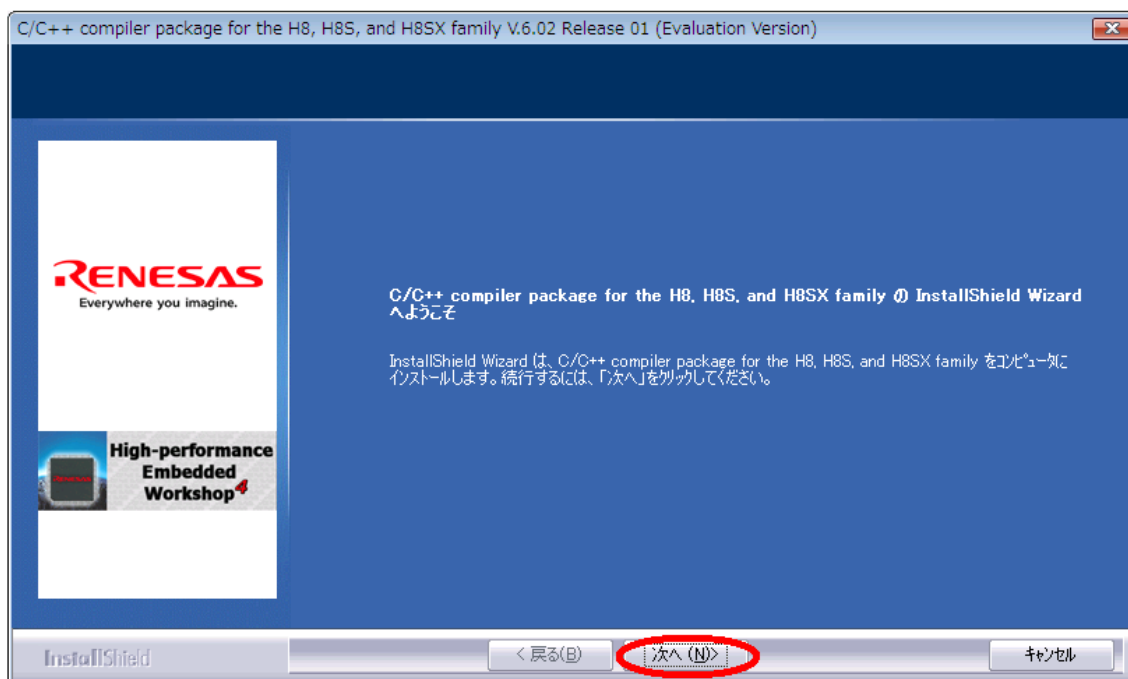



図 1-16 インストールウィザードが開始

手順⑦　規約に同意し, [はい] を押します.

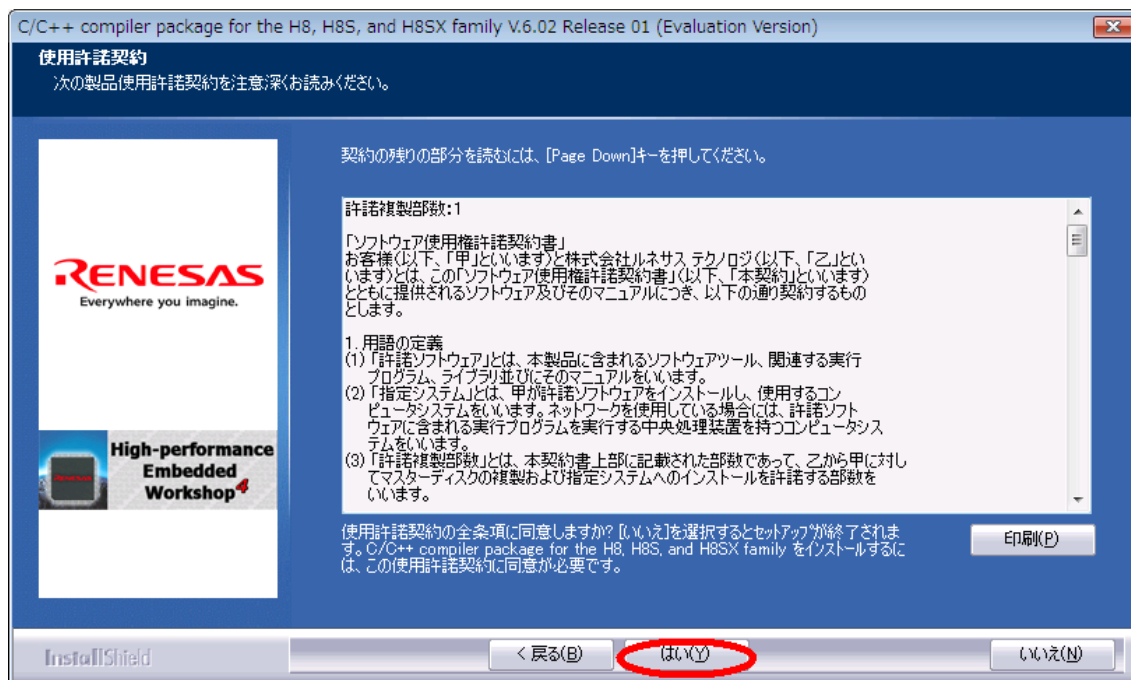



図 1-17　使用許諾規約に同意して進む

手順⑧　[その他の地域（日本，アジア)] をチェックし，[次へ] を押します.

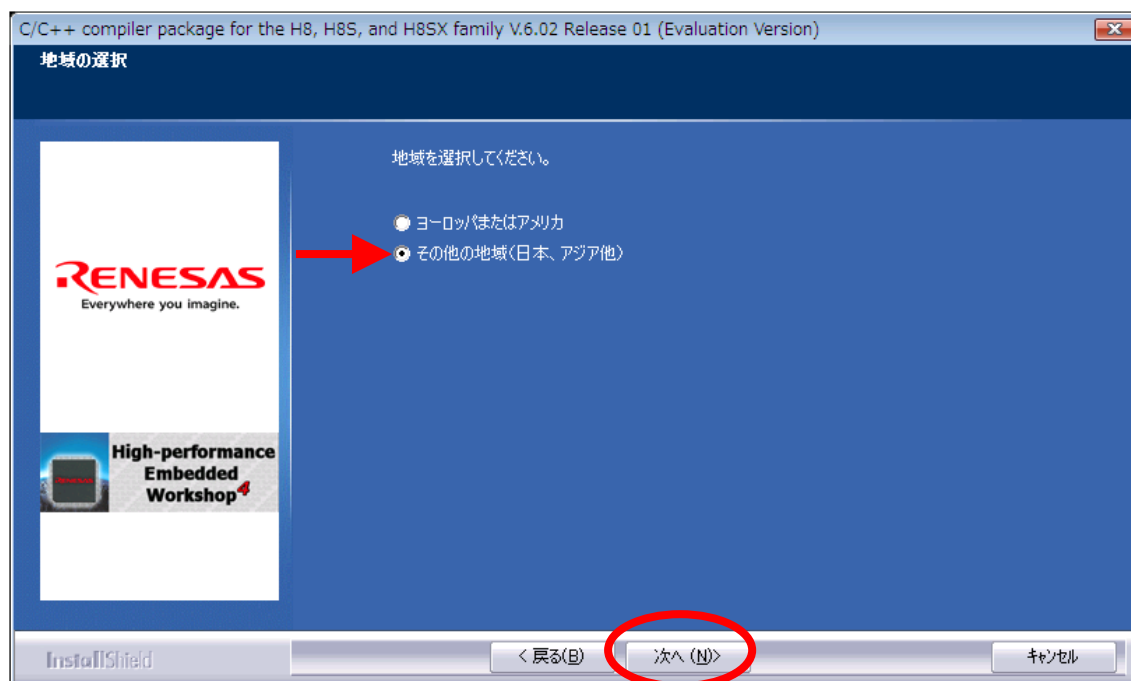



図 1-18　地域の検索

手順⑨　[インストール] を押すとインストールが開始されます.

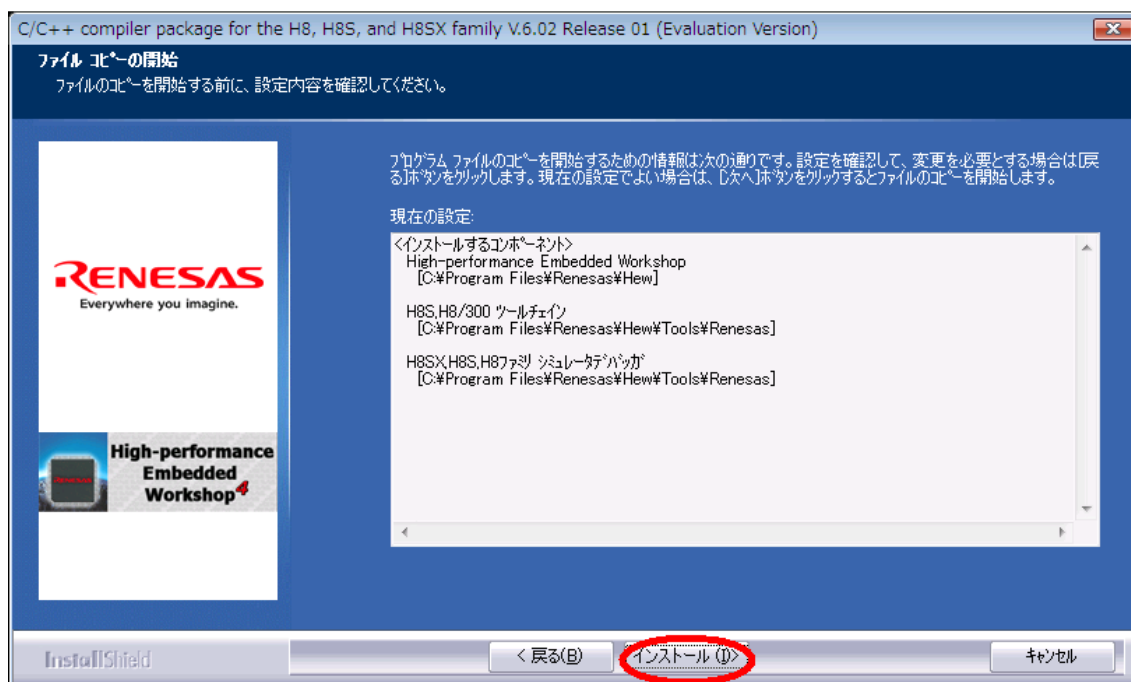



図 1-19　インストールの開始

手順⑩　インストールが終了すると以下の画面が表示されますので, [完了] を押してください.

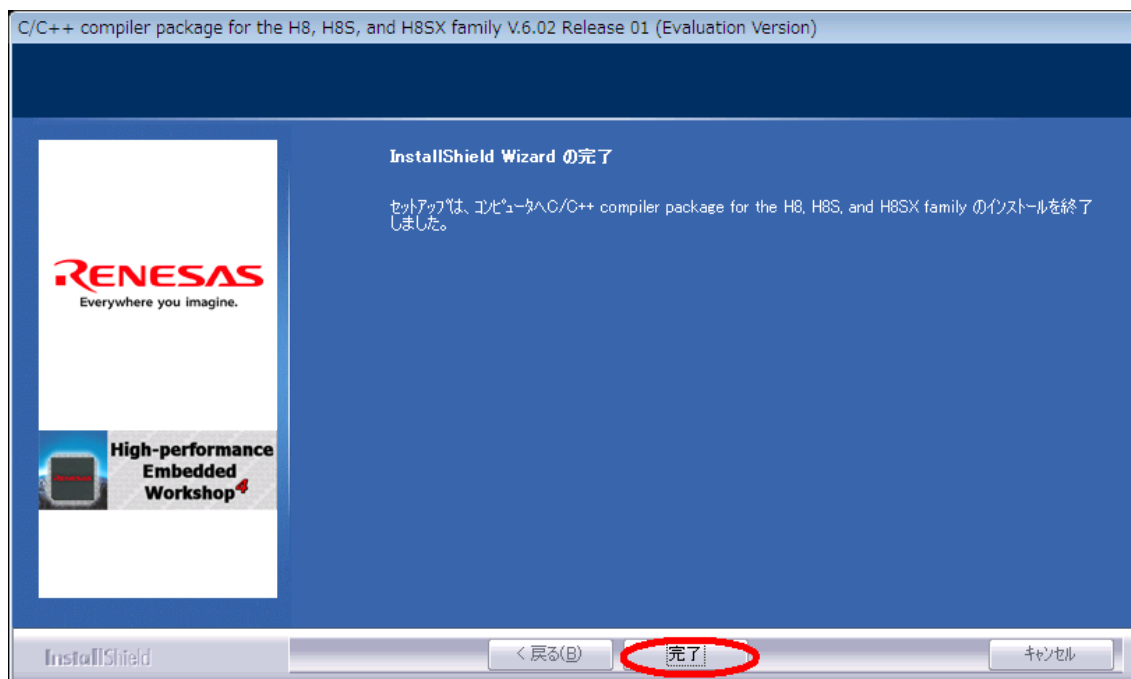



図 1-20　インストールの完了

手順⑪　インストール終了画面が出るので，［終了］を押してください．

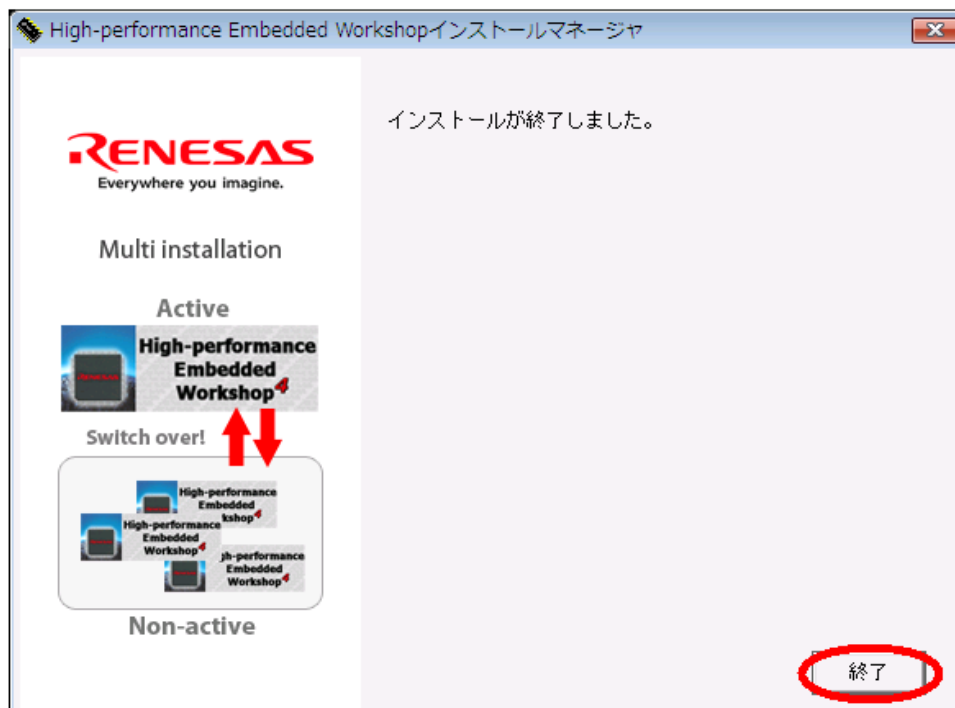



図 1-21　インストール終了画面

1.1.4.　FDT のインストール

次に CPU にファームウェアを書き込む祭に使用する FDT をインストールします．以下の手順に従ってインストールしてください．

手順①　ダウンロードしたインストーラ［fdtv***r**.exe］を起動します．
(※1:***にはバージョン,**にはリリース番号が入ります)
(※2:起動時に無償評価版の確認画面が表示されることがあります．その場合，OK を押すことで，②の画面が表示されます)

手順② [Next] を押します.

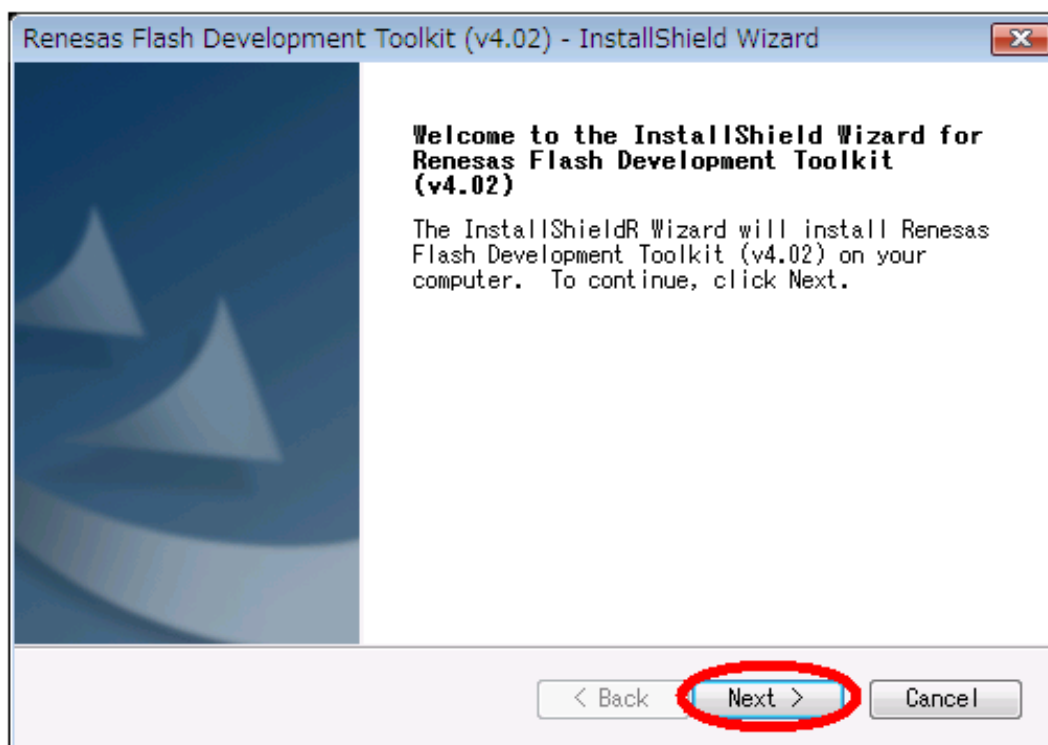



図 1-22 FDT のインストール開始

手順③ [Asia (Japanese)] を選択し， [Next] を押します.

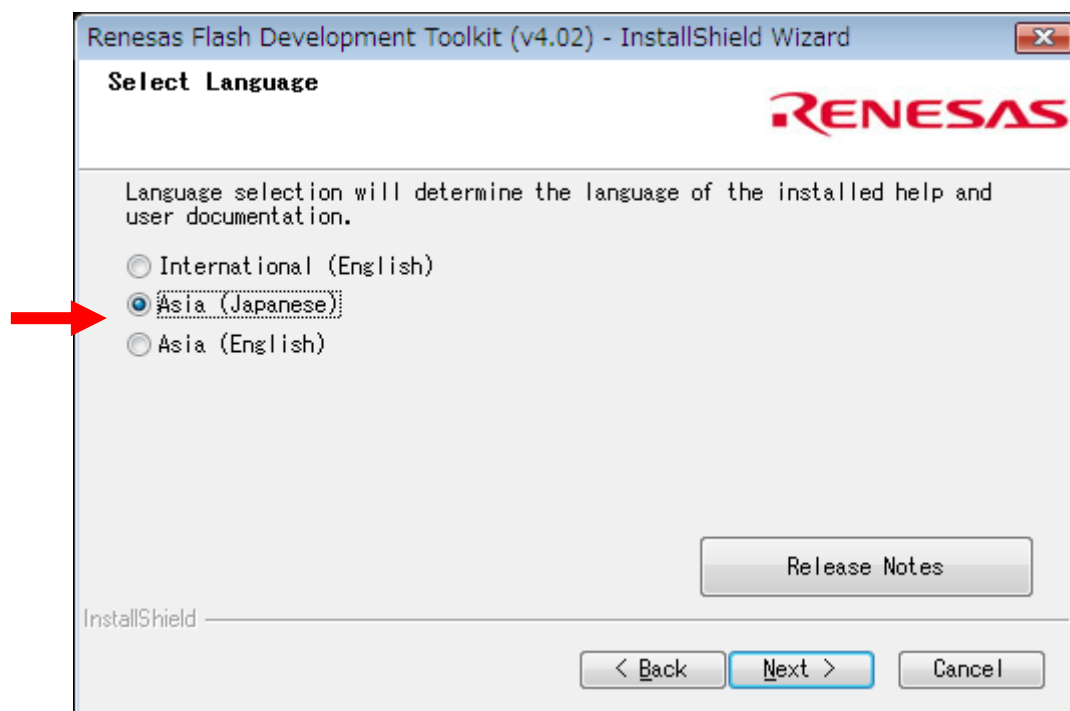



図 1-23 言語を選択

手順④　規約に同意し，［Next］を押します．

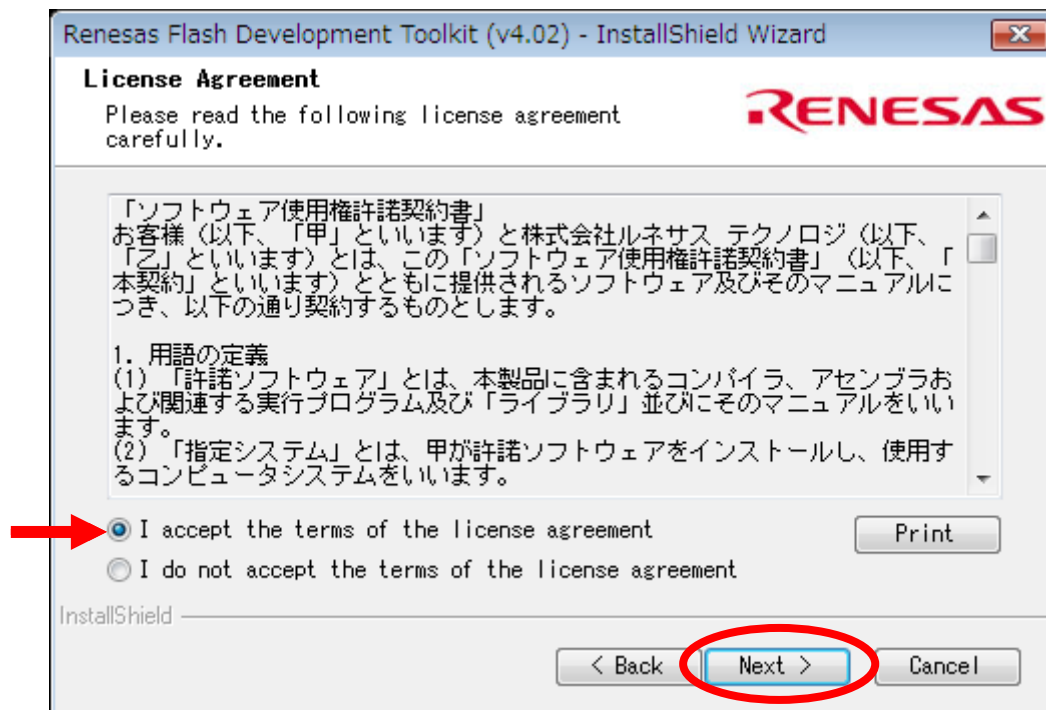



図 1-24　ソフトウェアの使用権許諾契約書に同意する

手順⑤　下のようにすべてにチェックをつけ，［Next］を押します．

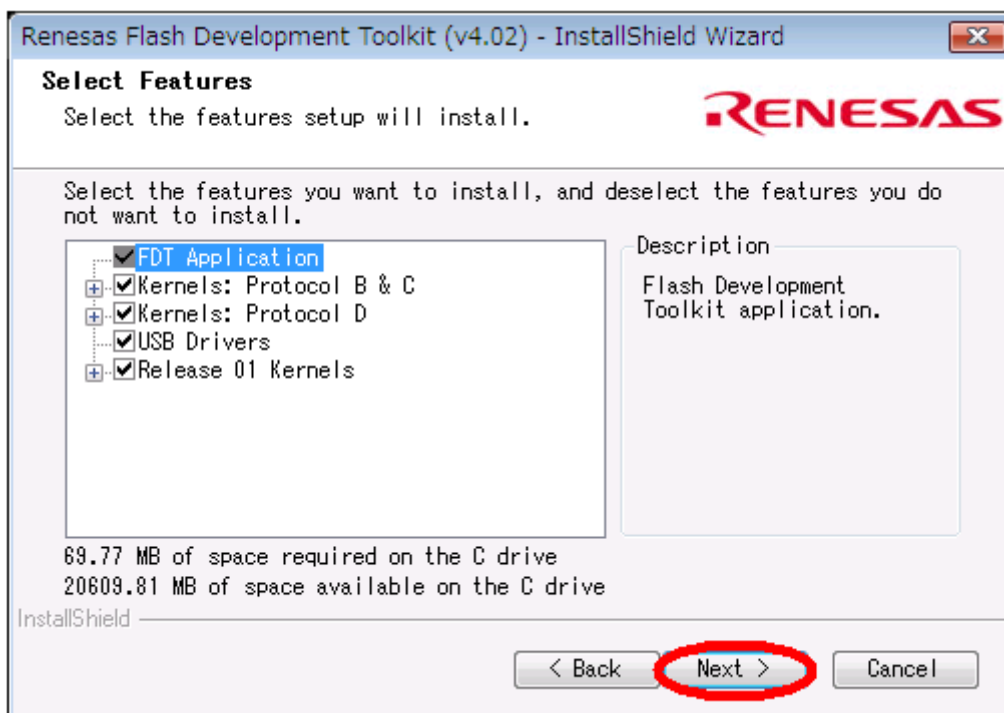



図 1-25　インストールするアプリケーションを選択

手順⑥　関連づけるファイル形式を指定し，[Next] を押します．

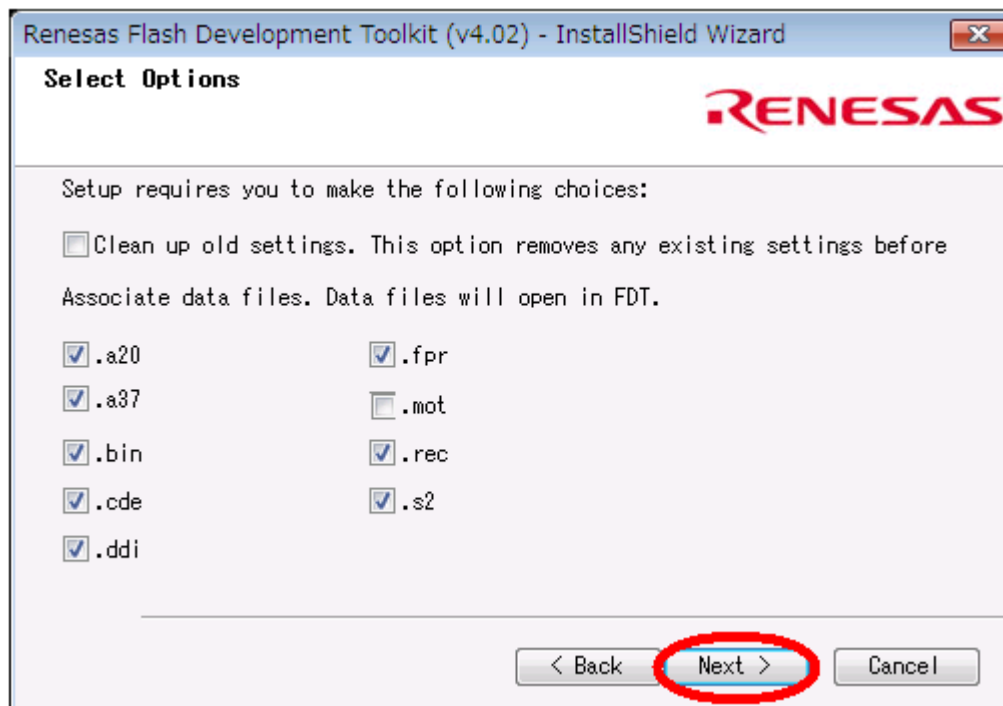



図 1-26　関連づけるファイル形式を指定

手順⑦　インストール先のフォルダ先を指定し， [Next] を押します．

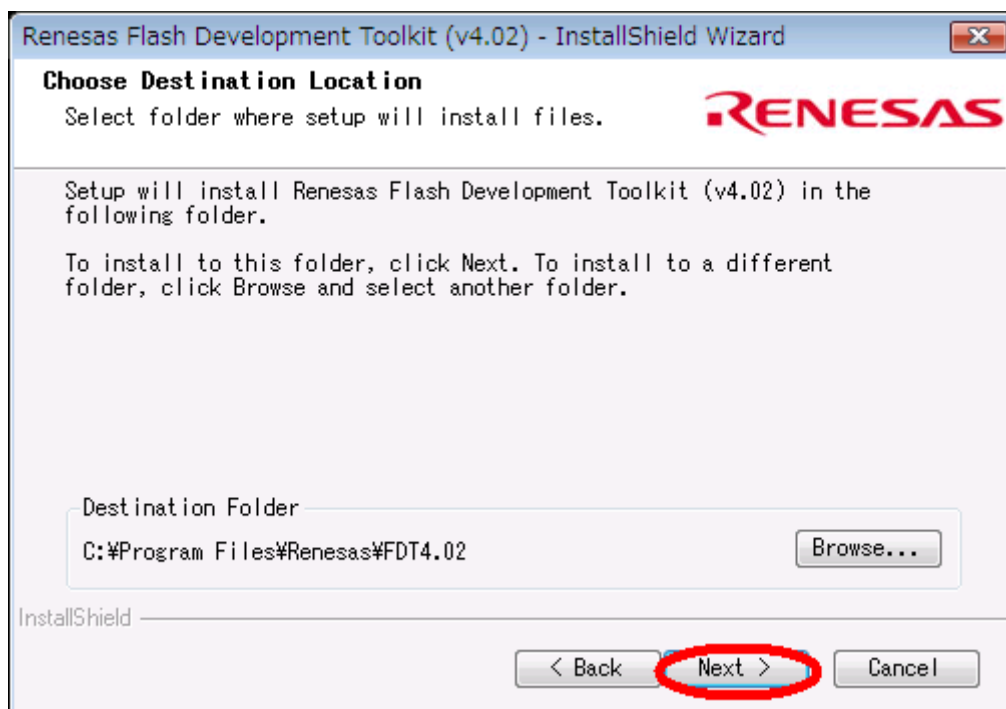



図 1-27　インストール先を選択

手順⑧　[Install] を押すと，インストールが開始されます．

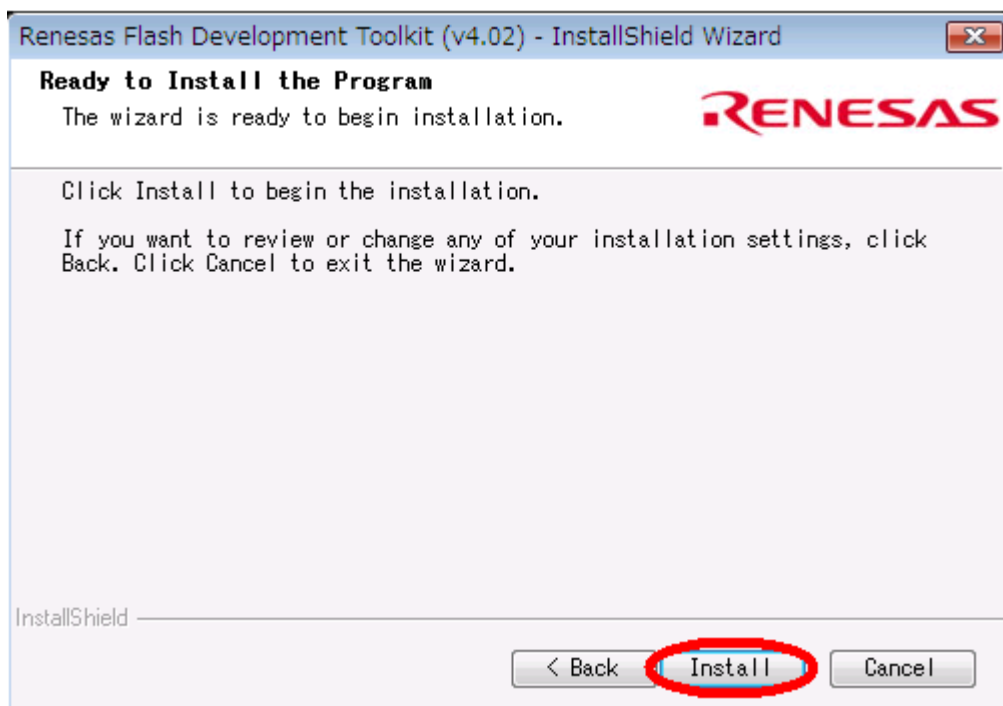



図 1-28　インストール開始

手順⑨　インストールが完了すると以下の画面が表示されますので，[Finish] を押しウィンドウを閉じます．

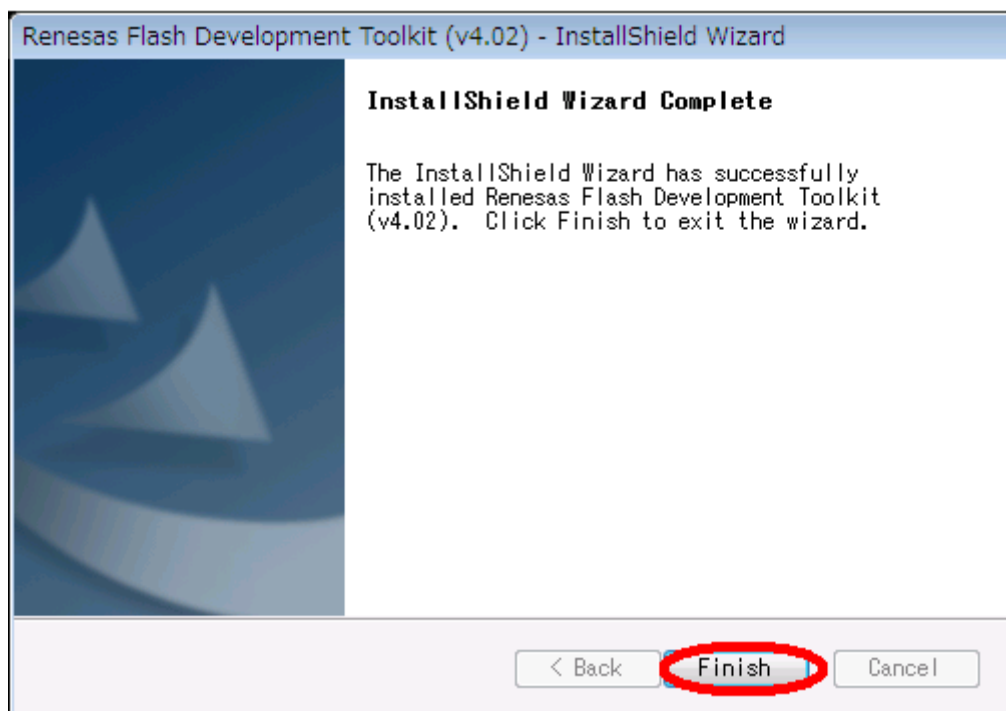



図 1-29　インストール完了

1.2. PC と CPU ボードの接続，書き込み

それでは実際に HEW を用いて，サンプルプログラムを CPU ボード「VS_WRC003」に書き込む方法について解説します．

ここでは次章で解説する LED を点灯させるサンプルプログラムの書き込みと実行しながらやり方を覚えましょう．

1.2.1. HEW の起動とサンプルプロジェクトの読み込み

手順①サンプルプロジェクトは以下の URL からダウンロードできます．以下の URL にまずはアクセスして下さい．

http://www.vstone.co.jp/top/products/robot/beauto/cdownload.html

手順② 「LED の点滅」ソース一式 をクリックして下さい．

H8マイコン統合開発環境「HEW」・C言語サンプルソース

本項目では、H8マイコン用C言語統合開発環境「HEW」のダウンロード、及びその環境に対応したBeautoCHaser/VS-WRC003用のC言語サンプルソースを公開しております。

●H8マイコン統合開発環境「HEW」インストールマニュアルのダウンロード
BeautoChaserに搭載しているCPUボード「VS-WRC003」用のプログラムを、ルネサルテクノロジ社の統合開発環境「HEW」を利用して開発するための導入ガイドです。
H8マイコンのC言語開発を行なうためのソフトウェアの導入について解説しています。説明中に登場するH8マイコン統合開発環境「HEW」及び関連ツールはこちらのwebページからダウンロードできます。

サンプルソースは以下よりダウンロードできます。

●「LEDの点滅」ソース一式　←「ＬＥＤの点滅」ソース一式をクリック
●「LEDの点滅」ビルド方法解説

●「無線コントローラVS-C1での操縦 」ソース一式
●「VS-WRC004によるサーボモータの制御」ソース一式

※VS-WRC003に接続するゲームパッドは、機種により正しく動作しない場合がありますのでご注意ください。弊社「VS-C1」は動作を確認しております。

(C)Copyright 2007 Vstone Co.,Ltd. All rights reserved.

図 1-30「LED の点滅」ソース一式画面

手順③ サンプルプロジェクト［WRC003_SampleProject］を［C:¥WorkSpace］に保存します．

以降，Windows のインストールされているドライブが C:の場合について記述します．ほかのドライブにインストールされている場合は，C:に置き換えてください．

手順④　保存が完了したら，「LED の点滅」ソース一式を適当な解凍ソフトで解凍し，ファイルの入ったフォルダ名を「WRC003_SampleProject_LED」にしておきます。HEW ではプログラムをワークスペースという単位で管理しています．

今回の WRC003_SampleProject_LED フォルダの中にはサンプルプログラムのワークスペースが入っています．このフォルダを C ドライブ下にある WorkSpace に入れてください．このフォルダの構成は［C:¥WorkSpace¥WRC003_SampleProject_LED］となります．

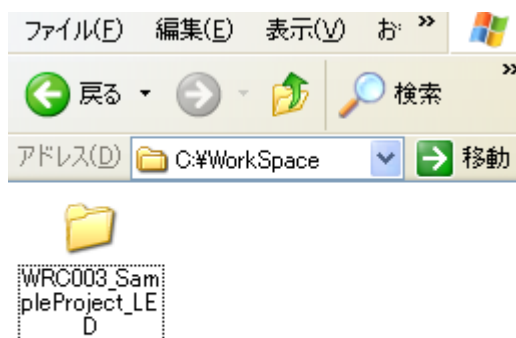



図 1-31　サンプルプロジェクトのフォルダ構成

手順⑤　Windows の［スタートメニュー］から［プログラム］> [Renesas] > [High-performance Embedded Workshop］を起動します．

起動時に“ようこそ”画面が表示されますが，キャンセルしてください．

手順⑥　メニューの［ファイル］から［ワークスペースを開く］を選択し，サンプルプロジェクトファイル内にある［WRC003_SampleProject.hws］を開きます．

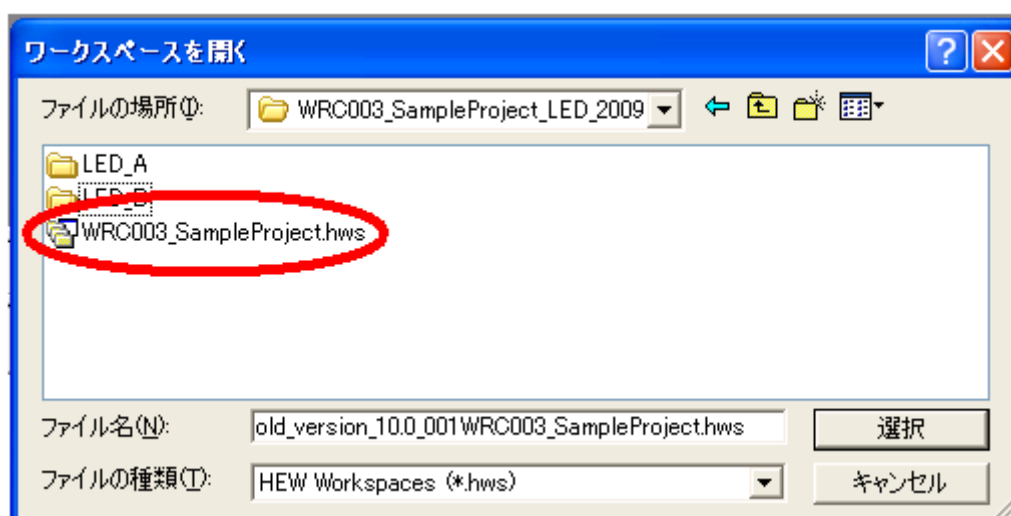



図 1-32　サンプルプロジェクトを選択

プロジェクトを開く際にツールチェインが異なる場合があり，その場合，変更を求められることがあります．変更する場合は，すべて OK を選択することで変更できます．

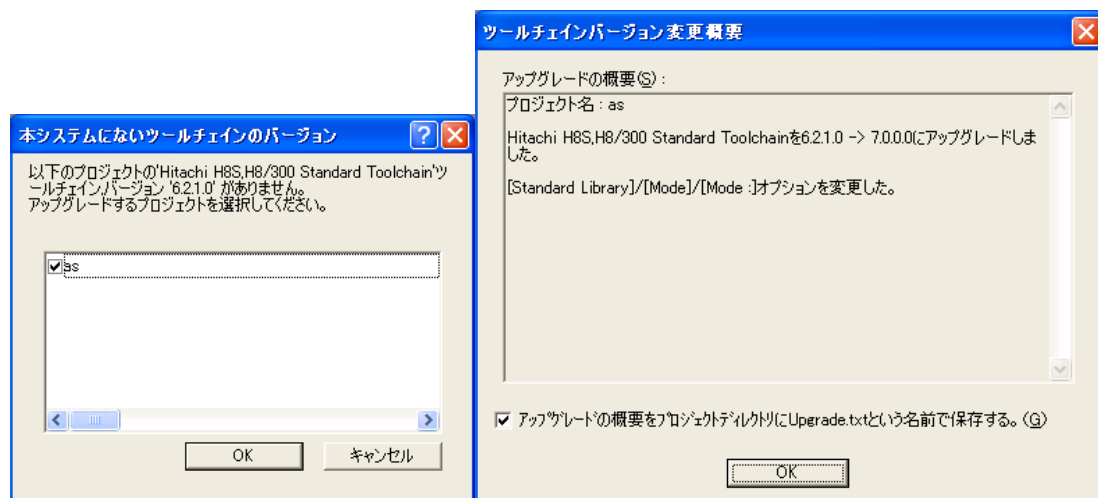



図 1-33　ツールチェインが異なる場合のエラー

手順⑦　ようこそ! というタイトルのダイアログが出てきますが，一度キャンセルボタンを押しましょう．

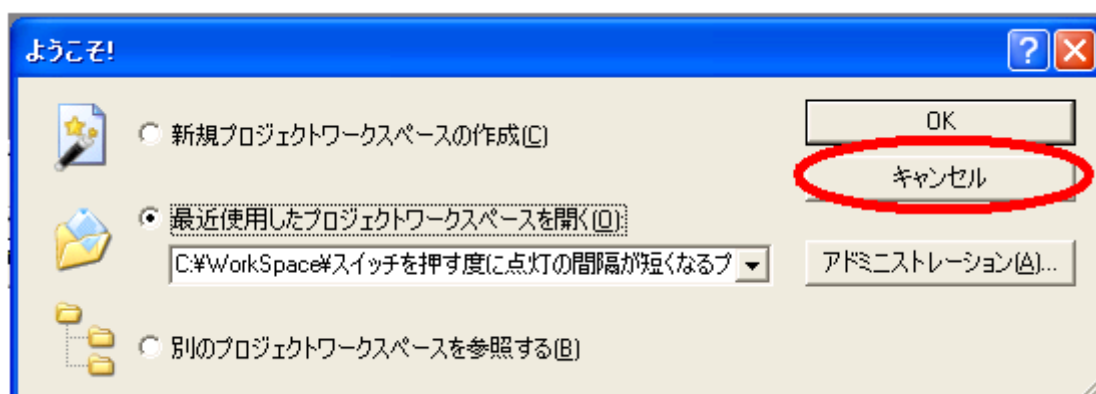



図 1-34　「ようこそ！」画面

手順⑧　すると HEW の初期画面が出てきます.

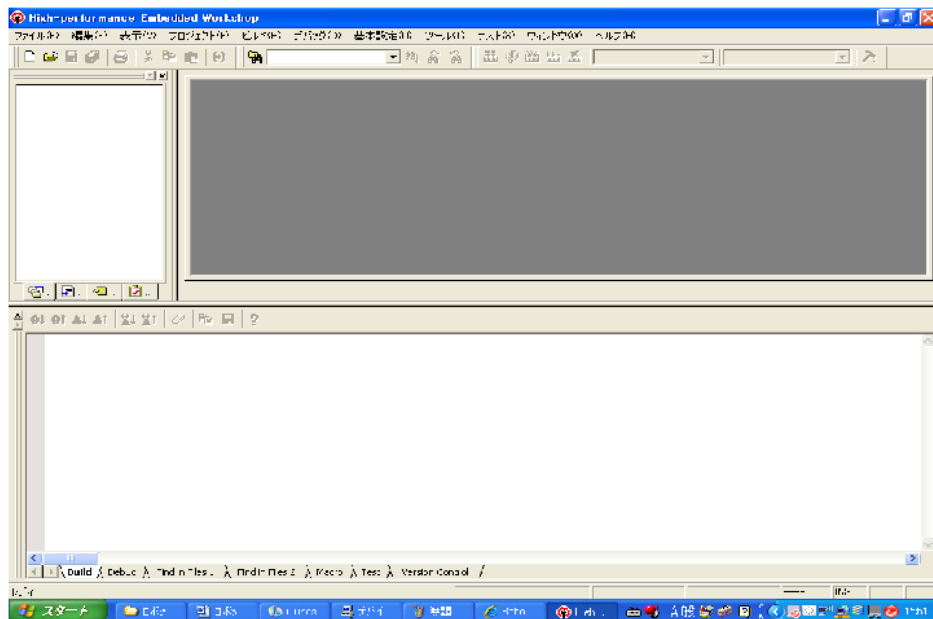



図 1-35　HEW 初期画面

手順⑨　メニューバーから「ファイル」→「ワークスペースを開く」を選ぶと選択画面になります.

C:¥WorkSpace¥WRC003_SampleProject_LED 内の WRC003_SampleProject.hws を選択して下さい.

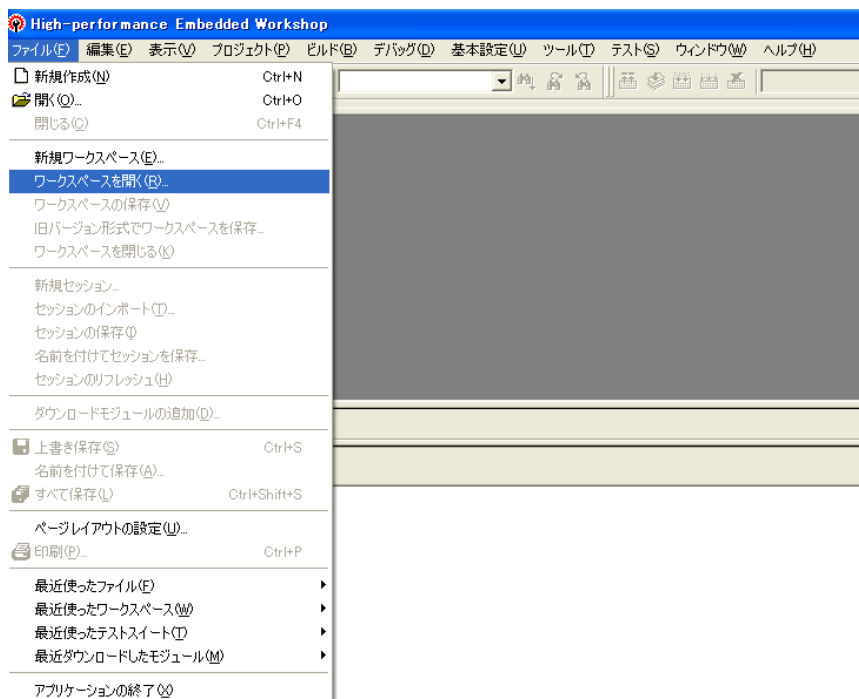



図 1-36　ワークスペースを開く画面

手順⑩　するとサンプルプログラムのワークスペースが読み込まれます．

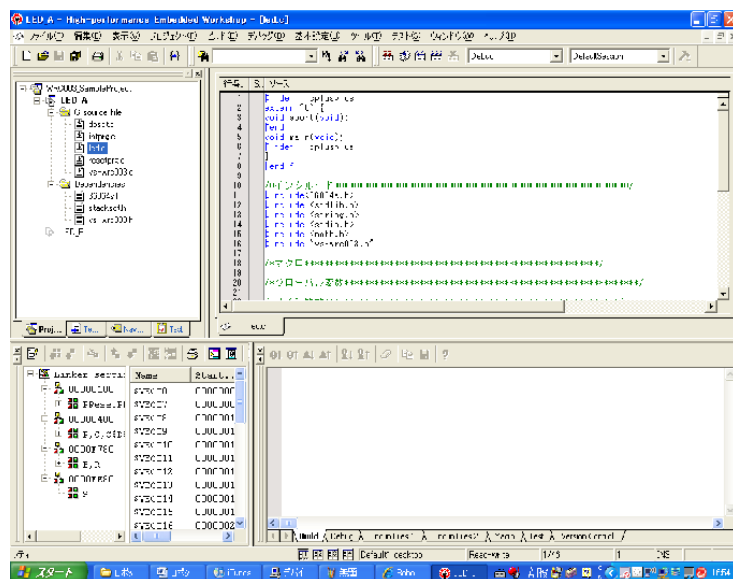



図 1-37　サンプルプログラムのワークスペース画面

手順⑪　では各エリアについて説明したいと思います．

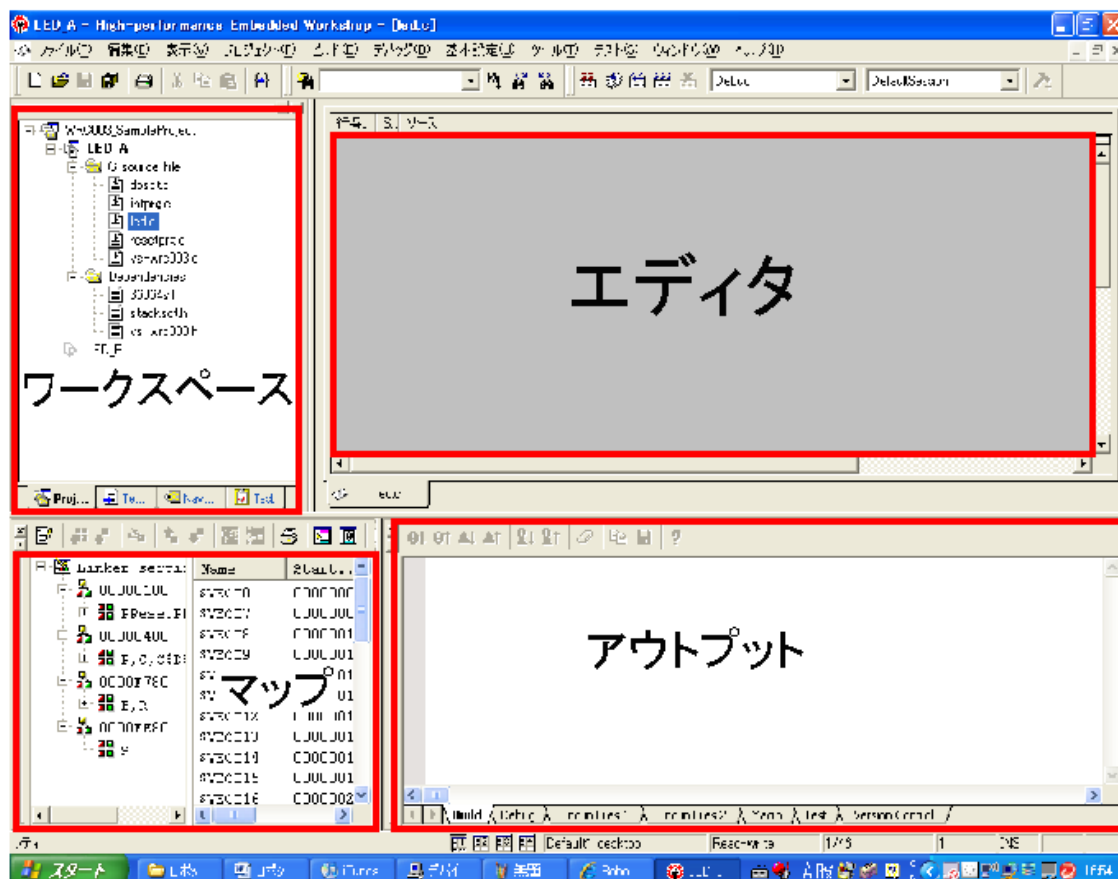



図 1-38　HEW の画面

（1）ワークスペース

現在開いているワークスペースに含まれるプロジェクト，ソースファイル，ヘッダファイルを階層構造で説明します．表示されているファイルはダブルクリックで開くことが可能です．また，右クリックでメニューからファイルの追加，削除なども行えます．

実際にプログラムを書き換えるのはソースファイルの［led.c］です．

（2）エディタ

各プロジェクト内のソース，ヘッダを編集できます．一般的なテキストエディタと同様のものですが，プログラムで使用する語句などに色がついたり，改行時に自動でタブを挿入してくれるなどプログラミングしやすいものとなっています．

（3）アウトプット

コンパイラ，リンカからのエラー，ワーニング（プログラムソースをコンパイラにかけた際に出力されるエラーメッセージ）などの出力や，文字列を検索した結果などが表されます．エラーや検索結果をダブルクリックすることで，そのエラーや文字列がある行がエディタに表示されることが出来ます．

（4）マップ

CPU 内の ROM（読み出し専用の半導体装置），RAM（書き込み専用の半導体装置）内にどのようなプログラム，データを配置したかを表示するものです．ROM，RAM を多く使う場合，入りきらなくなることがありますので，そういった場合のチェックに使用します．本書のプログラムでは使用することはありません．

それでは，ワークスペースを見てみましょう．サンプルプログラムのワークスペースには LED_A と LED_B の二つのプロジェクトが含まれており，各プロジェクトには以下のようなファイルが含まれています．複数のプロジェクトがある場合，ワークスペース内の［LED_A］を右クリックし，プロジェクトをアクティブにしてください．

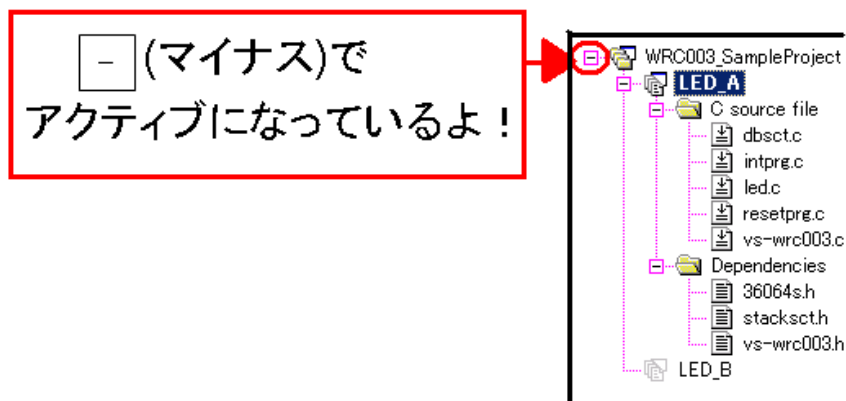



図 1-39　ワークスペースの「LED_A」をアクティブにする

プロジェクト内の説明をします.

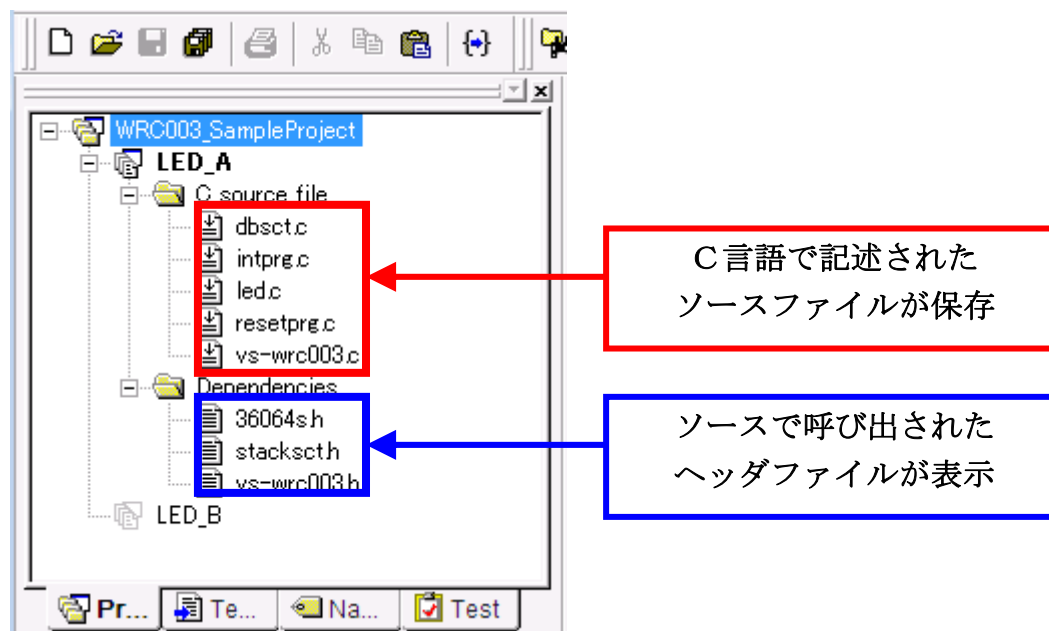



図 1-40 「LED_A」のプロジェクトの内容

各フォルダにある重要なファイルについて説明します.

(1) C souce file

・led.c

led.c は LED_A プロジェクトのメインのソースファイルで，main () 関数が含まれるファイルです．今回は main () 関数を中心にプログラミングを進めていきます.

・Intprg.c

Intprg.c は割り込み処理について記述されたソースファイルです.

・vs-wrc003.c

vs-wrc003.c は「VS-WRC003」を使用するための関数が入っているソースファイルです.

(2) Dependencies

・36064.h

36064.h は「VS_WRC003」に搭載されている CPU の機能について定義したファイルです.「VS_WRC003」でプログラムする場合は必ず読み込む必要があります．これを読み込まないとプログラムが正常にうごきません.

・vs-wrc003.h

vs-wrc003.h は「VS_WRC003」用の関数を定義したファイルです．ここには関数名，関数の説明が記載されています.

1.2.2. インクルードファイルディレクトリの設定

開いたプロジェクトに前項で説明した「Dependencies」に必要なヘッダファイル（HEWではインクルードファイル）は揃っていない人もいると思います．ヘッダファイルは「#include」で読み込みますが，HEW ではどのようなフォルダから読み込みか設定する必要があります．

以下の手順に従って，HEW でインクルードファイルディレクトリを設定しましょう．

手順①メニューの［ビルド］内の［H8S, H8/300 Standard Toolchain..］からツールチェイン（コンパイラ，リンカなどのこと）を設定します．

図 1-41　メニューの［ビルド］からツールチェインの設定を行う

Key word　ツールチェイン

ツールチェインとはプログラムを製作するのに使われるエディタ，コンパイラ，リンカなどのツールの集合体，エディタの出力がコンパイラの入力になり，コンパイラの出力がリンカの入力になる．このように連鎖的に使われることからツールチェインと呼ばれる．

手順②　ツールチェインの設定を開くと以下のようなウィンドウが表示されます．

ここで，［コンパイラ］タブ内のカテゴリで［ソース］，オプション項目で［インクルードファイルディレクトリ］を選択すると，以下の画面が表示されます．この画面で，どこからヘッダファイルを読み込むかを設定することができます．［追加］を押し，新しく読み込むフォルダを設定します．

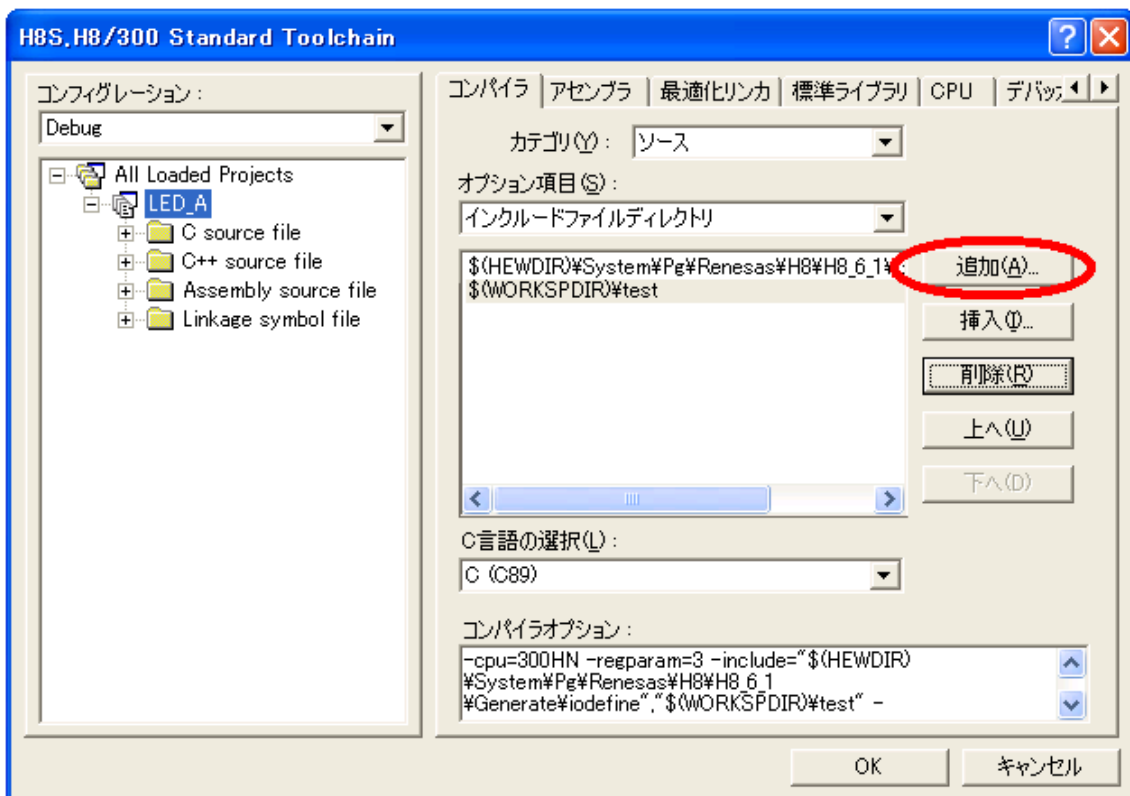



図 1-42　インクルードファイルを読み込むディレクトリを設定

手順③　インクルードディレクトリ追加用のウィンドウが開いたら，相対パスに［Costom directory］を選択し，ディレクトリに各 CPU 用のインクルードファイルがある［iodefine フォルダ］のパスを設定し，［OK］を押します．

なお，インストール時インストール先のフォルダを変更していない場合，［indefine フォルダ］は以下のようになります．

[C:¥Program Files ¥Renesas ¥Hew ¥System ¥Pg ¥Rnesas ¥H8 ¥H8_*_*¥Generate ¥iodefine]

(*にはツールチェインのバージョンの数字が入ります．例：H8_7_0)

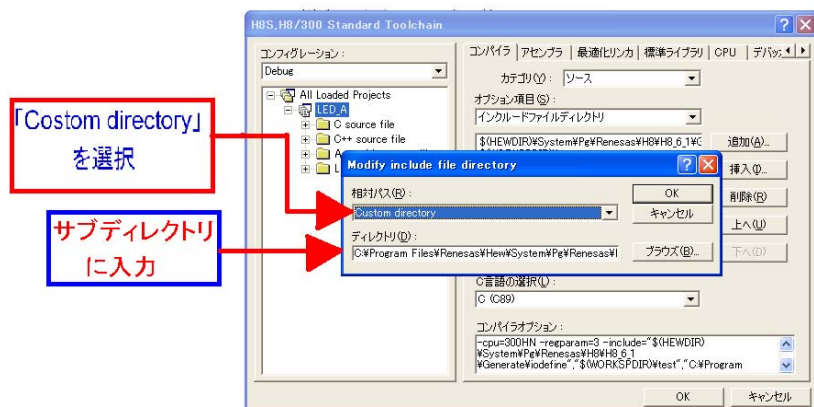



図 1-43　インクルードファイルのあるフォルダを設定

手順④　設定したフォルダが追加されていたら［OK］を押して，ツールチェインの設定を終了します．

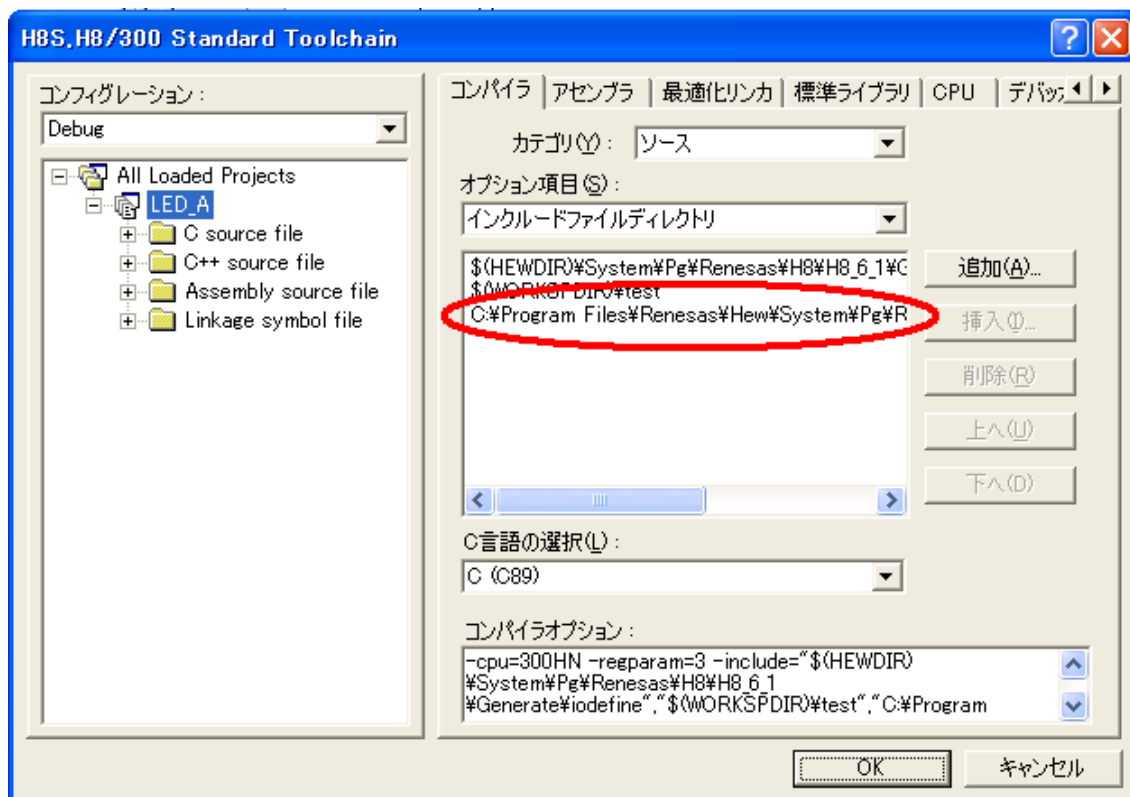



図 1-44　設定したフォルダが追加される

1.2.3.　　ビルドとヘッダファイルの編集を行おう

ここで，ビルドを行いCPUボードに書き込むためのファイルを生成します．ビルドはメニューの［ビルド］内の［すべてのビルド］を選択すると行えます．2回目以降は［ビルド］内の［ビルド］を選択またはF7キーを押すことでビルドできます

Key word　ビルド

　作成したソースコードに対してリンクやコンパイルの作業を行い，プロジェクトから実行可能なプログラムを作成することです．

手順①　メニューバーから［ビルド］→［すべてをビルド］を選んで下さい．

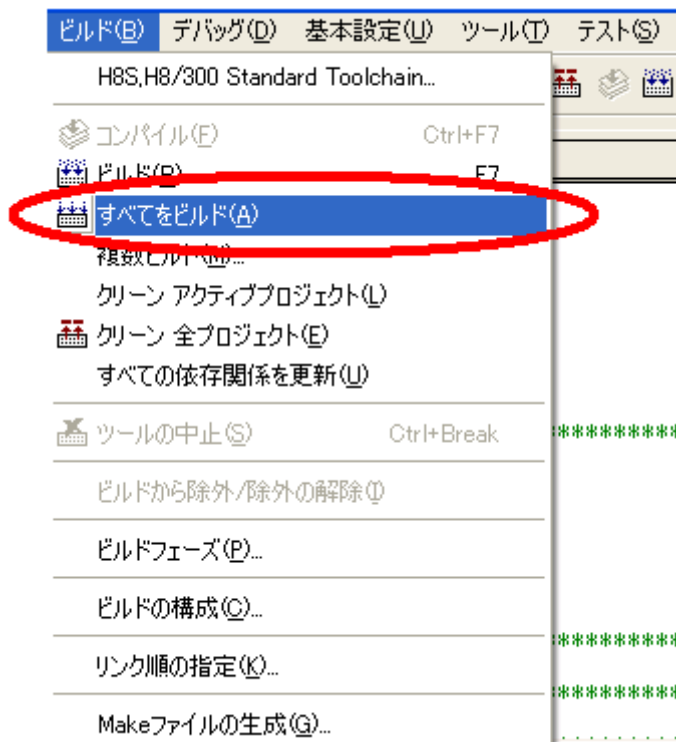



図 1-45　メニューの［ビルド］から［すべてをビルド］を選択

手順②　ビルドを行うとアウトプットウィンドウにエラーが表示されます．エラーの原因がどこか探したいときは，エラーの行をダブルクリックすると画面上のエディタにエラーが表示されます．最初にエラーが出ている行を見てください．ここから今回 36064s.h というファイル（これは先ほどメニューバーの「ビルド」→「H8S, H8/300 Standard Toolchain...」から追加したファイルになります）にエラーがあることがわかります．この行をダブルクリックすると，HEW 画面上のエディタ部分に 36064s.h が開かれます．

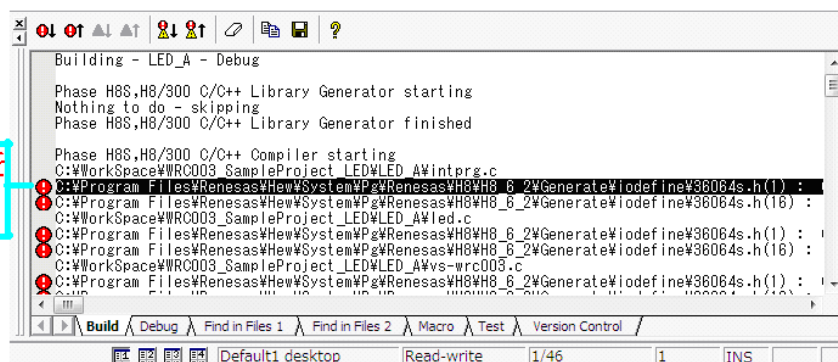



図 1-46　アウトプットウィンドウに表示されたエラー

手順③　エディタに 36064.h が表示されます．この 1 行目にコメントアウトされていない一文がありますので，[//] を入力してコメントアウトするか，1 行目を削除して下さい．

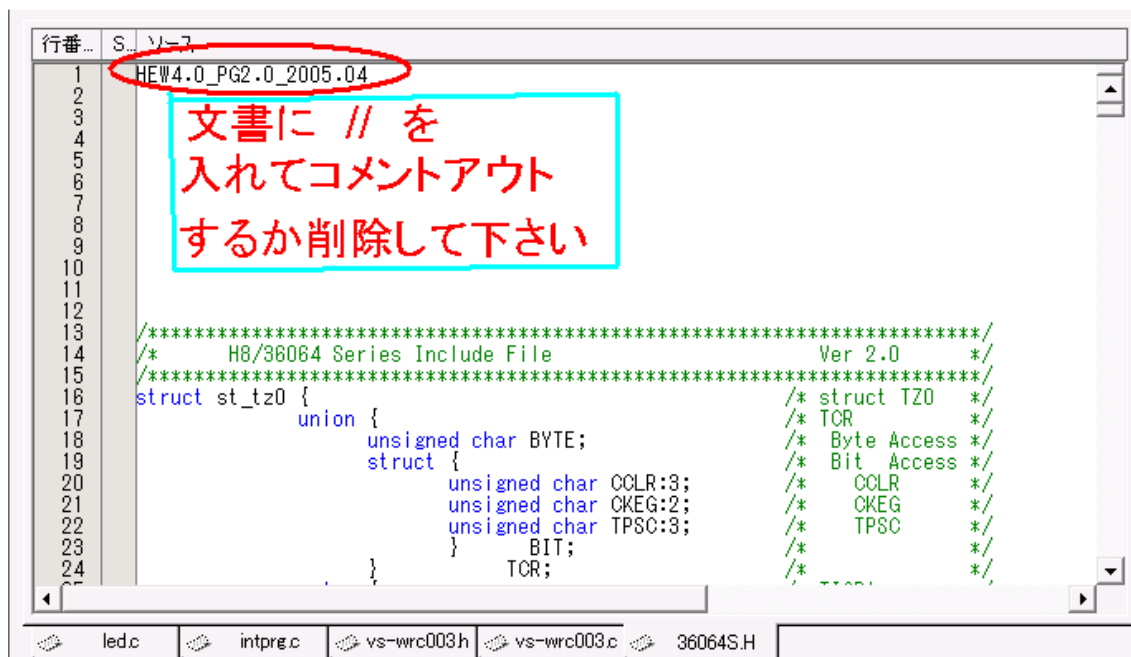



図 1-47　プログラムのエラー部分

手順④　それではもう一度ビルドしてみます．今度はエラーが出なかったはずです．右下に Build Finished 0 Errors,，0 Warnings と表示されているのを確認してください．

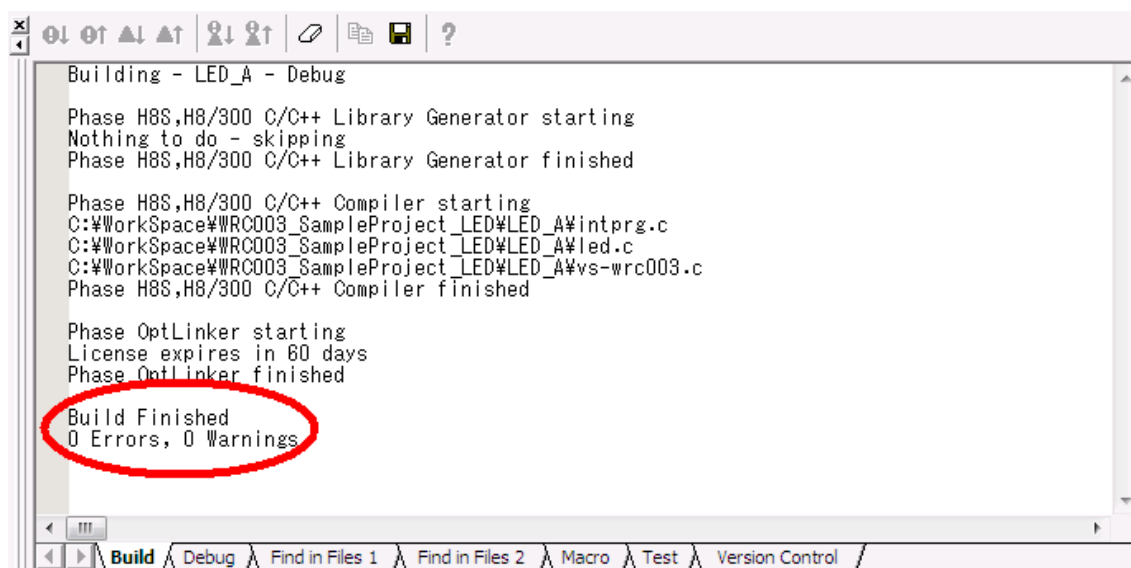



図 1-48　ビルド完了画面

1.3. CPU ボードへのプログラムの書き込み

それでは，先ほど生成した mot ファイルを FDT で CPU ボードに書き込んでみましょう．

今回使用する CPU ボードである VS_WRC003 には USB ボードである VS_WRC003 には USB シリアル変換 IC が内臓されているので，書き込みには COM ボードを使用します．まず，USB シリアル変換 IC のドライバをインストールする必要があります．

手順①　以下のサイトにある BeautoBuilderNEO 取扱説明書をダウンロードして下さい．

http://www.vstone.co.jp/top/products/robot/beauto/cdownload.html#builder

Beauto Chaser専用ソフトウェア「BeautoビルダーNEO」

BeautoビルダーNEOは、Beauto ChaserをPCと接続してより高度なプログラミングを行なうためのソフトウェアです。現在の最新版は以下よりダウンロードできます。
※車輪等ロボットの構成の違いにより、Beauto Balancerではご利用いただけません。

■公開日　2009/7/16
■バージョン　Release3 (製品バージョン1,0,0,3)
■追加/変更点　・設定エリアで「センサ3」「センサ4」が逆になっていた点を修正
※新しいバージョンのBeautoビルダーをインストールする場合は、こちらの説明に従ってデータのバックアップ、及び古いバージョンのアンインストールを行なってください。
■ファイル　BeautoBuilderNEO_Inst_003.exe
■USBシリアルドライバ　CP210x_VCP_Win2K_XP_S2K3.exe
※PCとBeautoが通信するために必要となりますので、必ずダウンロードしてPCにインストールしてください。
■取扱説明書　BeautoBuilderNEO_docs.zip
クリックして
取扱説明書をダウンロード
■主な更新履歴
Release2 (製品バージョン1,0,0,3)　・設定エリアで「センサ3」「センサ4」が逆になっていた点を修正
Release2 (製品バージョン1,0,0,2)　・シリアルポート番号を自動検出する方式に変更
Release1 (製品バージョン1,0,0,1)　・初版

図 1-49　取扱説明書ダウンロード画面

手順②　取扱説明書内の「1-2-2 USB シリアルドライバのインストールの手順」に従ってドライバをインストールしてください．

手順③　ドライバのインストールが完了したら，CPU ボードと PC を USB ケーブルで接続します．この状態で，デバイスマネージャ(※)を開き COM ポートの番号を確認します．(下の画面では COM４)．このとき，COM９以上のポートが使用されている場合は，COM８以下のポートに変更してください．

※ デバイスマネージャはマイコンピュータを右クリックしプロパティを開き，[ハードウェア］タブ内の［デバイスマネージャ］を押すと表示されます．

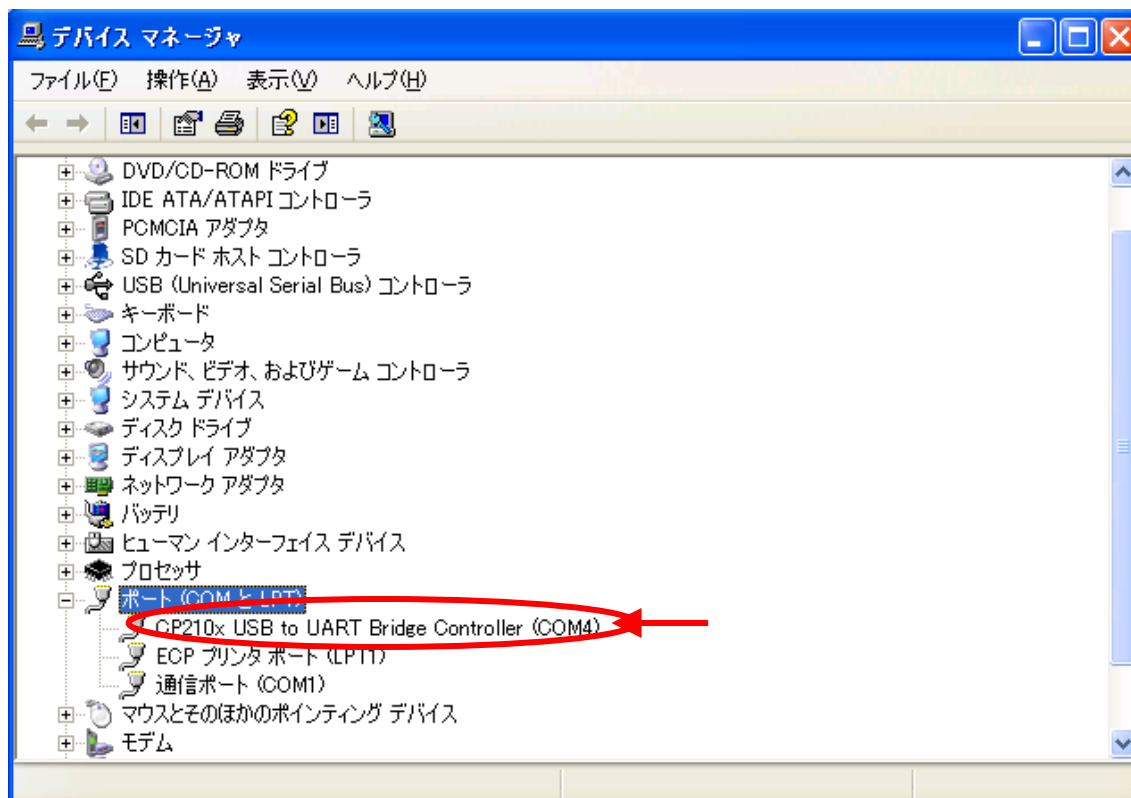



図 1-50　デバイスマネージャで COM ポートの確認

手順④　次に FDT を開きます．スタートメニューの［プログラム］> [Renesas] > [Flash Development Toolkit*.**] > [Flash Development Toolkit*.**Basic] を開きます．
(*.**はバージョンの番号)

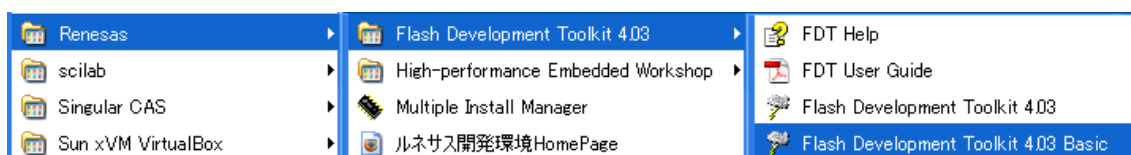



図 1-51　FDT を確認

手順⑤　初期起動時は, 以下のオプションが表示されます. 2 回目以降は表示されませんが, ツールバーの［オプション］>［新規設定］から同様の設定を行うことも出来ます.

設定中は，CPU ボードから USB ケーブルを抜かずに PC と接続し、電源をオンにしたままにしてください.

はじめの画面で，CPU ボードに利用されている CPU の種類を設定します．フィルタに 36064 と入力すると H8/36064F が出てきますので，これを選択し［次へ］を押しましょう

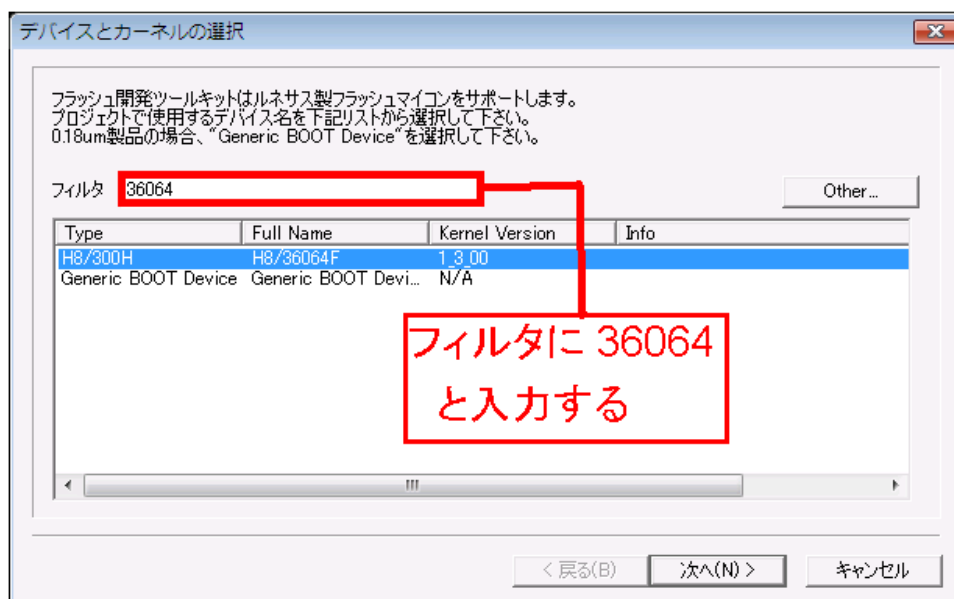



図 1-52　CPU の種類を選択

手順⑥　通信ポートに Beauto Chaser がつながっている COM ポートを設定し,［次へ］を押します.

図 1-53　COM ポートの選択

手順⑦　CPU の動作周波数を設定します．今回使う CPU ボードでは 14.7456MHz の推奨発信機が実装されていますので，[入力クロック] を [14.7456] に設定し，[次へ] を押します．

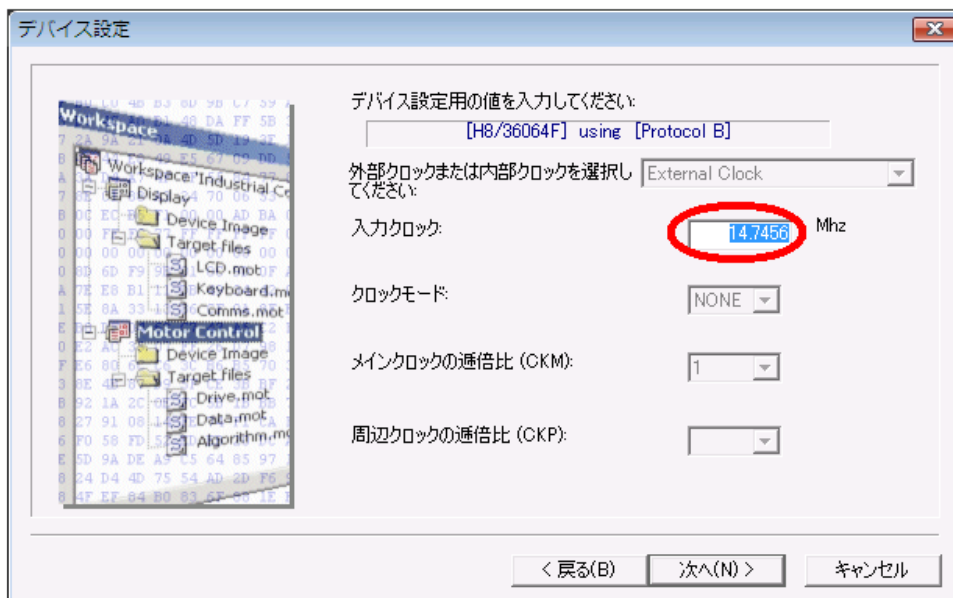



図 1-54　CPU の動作周波数を設定

手順⑧　次に接続の設定をします．書き込みはブートモードを使用しますので，以下の画面（接続タイプ）にて，[BOOT Mode] を選択し，ボーレートはそのままの設定で [次へ] を押します．ここでは何も変更しなくてかまいません．

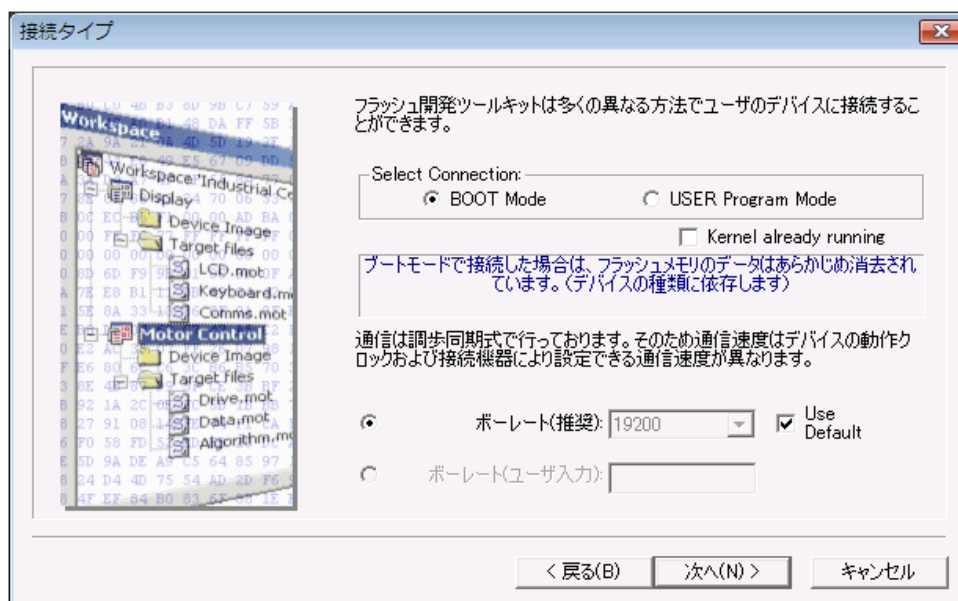



図 1-55　接続タイプの設定

手順⑨　書き込みオプションでは, [Protection] は上書きする際に自動的に消去を行ってくれる [Automatic] を選択, 出力メッセージは細かく表示するために [Advanced] を選択し [完了] を押します. ここでも何も変更しません.

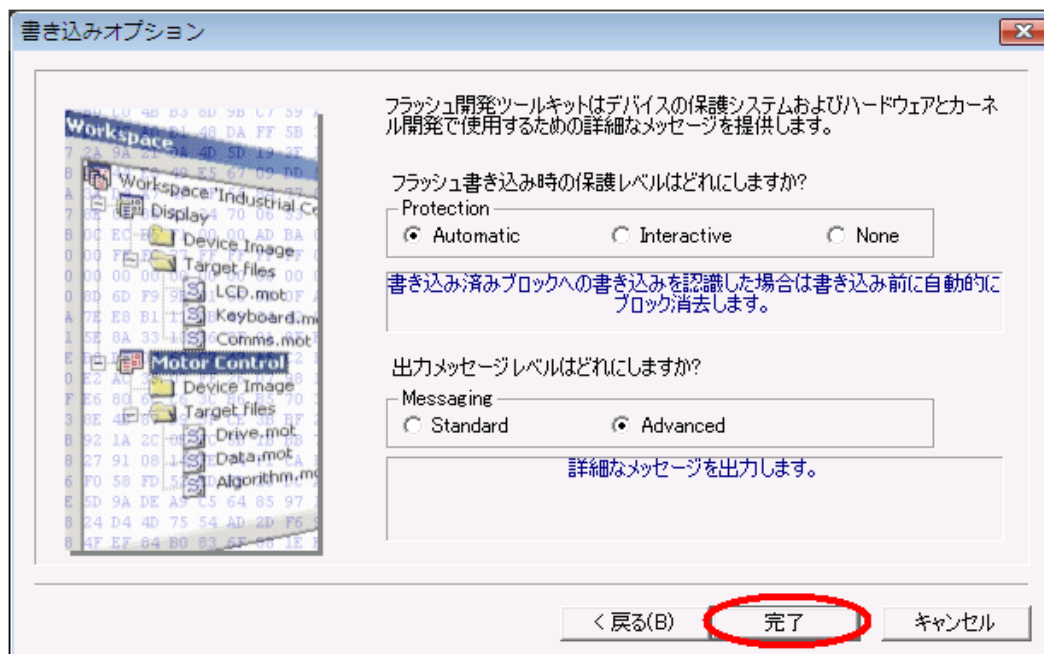



図 1-56　フラッシュ書き込み時の保護レベルの設定

以上で設定は完了です.

1.3.1.　実際に CPU ボードにプログラムを書き込んでみよう

手順①　[User/DataArea] をチェックし，右端の[▶] ボタンを押します.

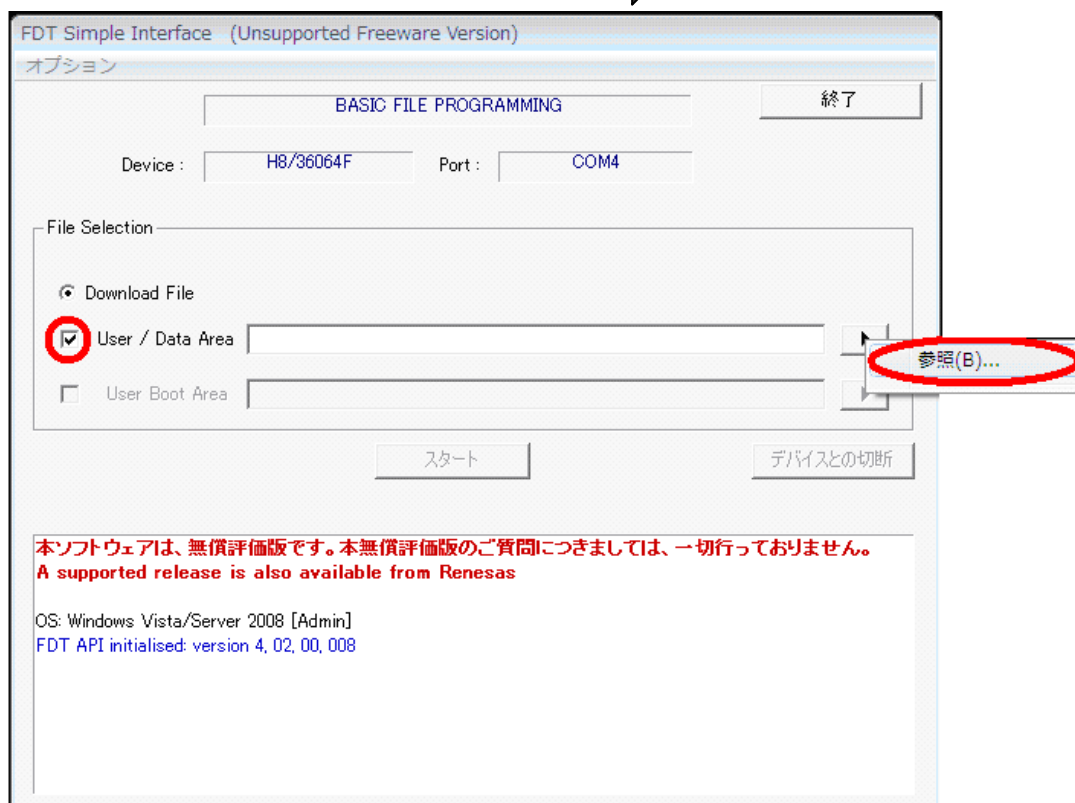



図 1-57　mot ファイルの保存先を選択

手順②　今回は，サンプルプロジェクトの[（プロジェクトの保存先，デフォルト）¥WRC003_SampleProject¥LED_A¥Debug] 内の [LED_A.mot] を選択します.

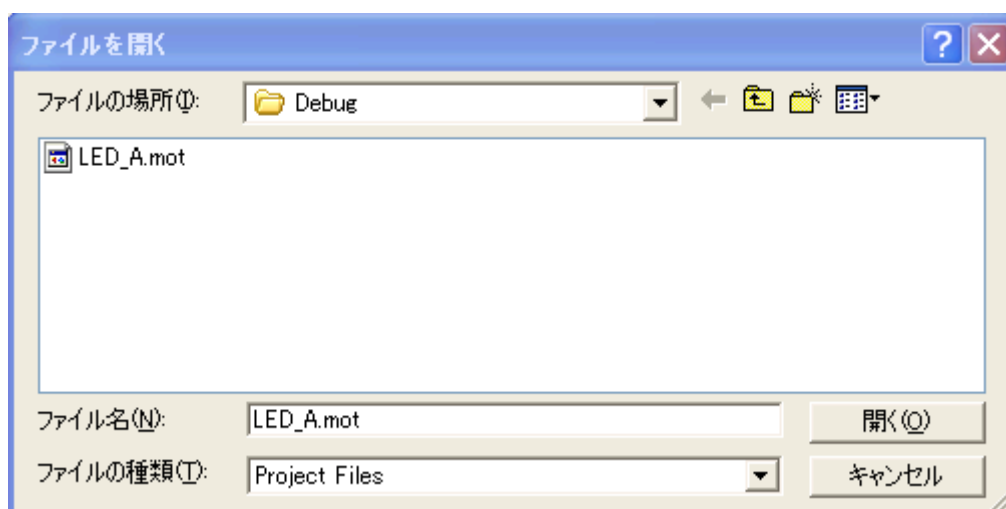



図 1-58　[LED_A.mot] を選択

手順③　CPU ボードの電源スイッチを切ります.

手順④　スイッチを押します.

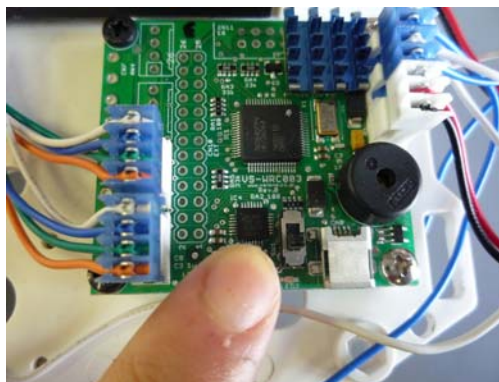



図 1-59　　CPU ボードのスイッチを押す

手順⑤　スイッチを押したまま USB コネクタを差し込みます.

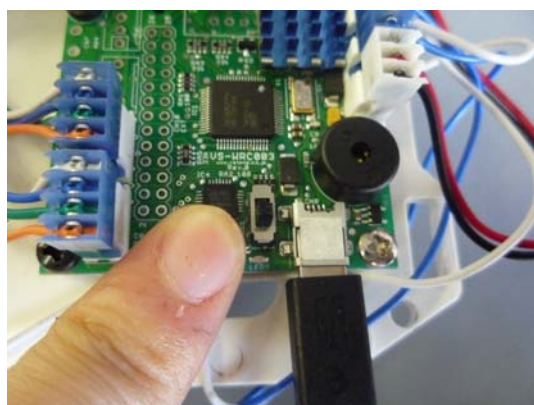



図 1-60　スイッチを押したまま USB コネクタを差し込む

手順⑥　スイッチを押したまま，電源スイッチを押します.

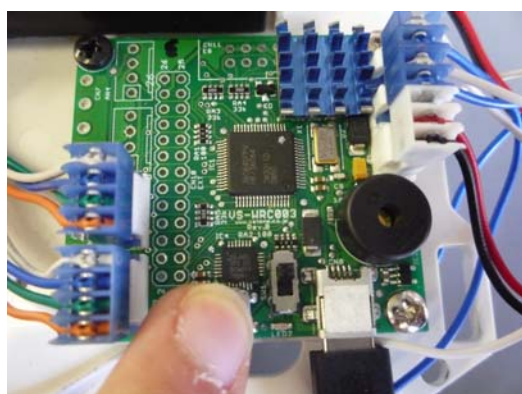



図 1-61　スイッチを押したまま電源スイッチを入れる

手順⑦　この状態のまま，FDT のスタートボタンを押します．書き込み中は電源スイッチから手を離さないで下さい．

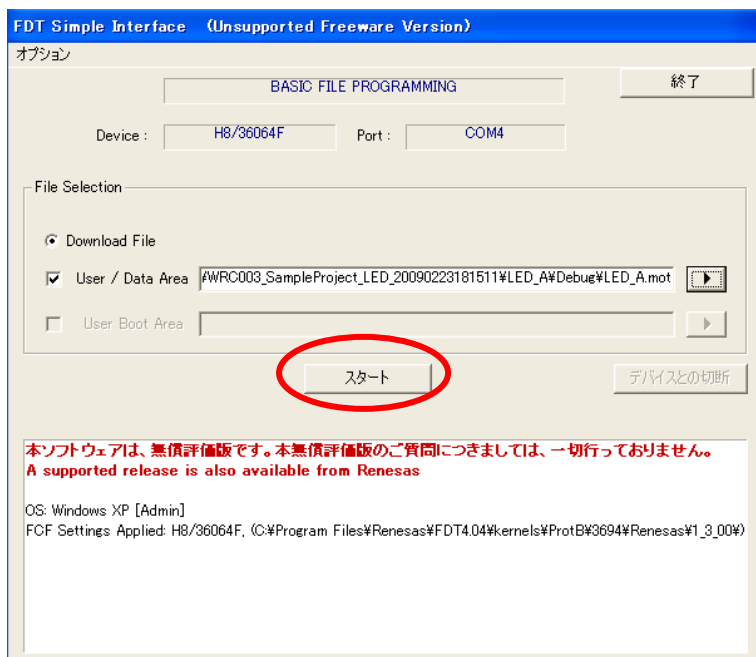



図 1-62　CPU のスイッチを押したまま，FDT のスタートボタンを押す．

手順⑧　下の図のように［書き込みが完了しました］と表示されれば，書き込み完了です．表示されない場合は［デバイスの切断］を押した後，CPU ボードから USB ケーブルを抜き，再度書き込みの手順とおりにはじめから書き込みを行ってください．

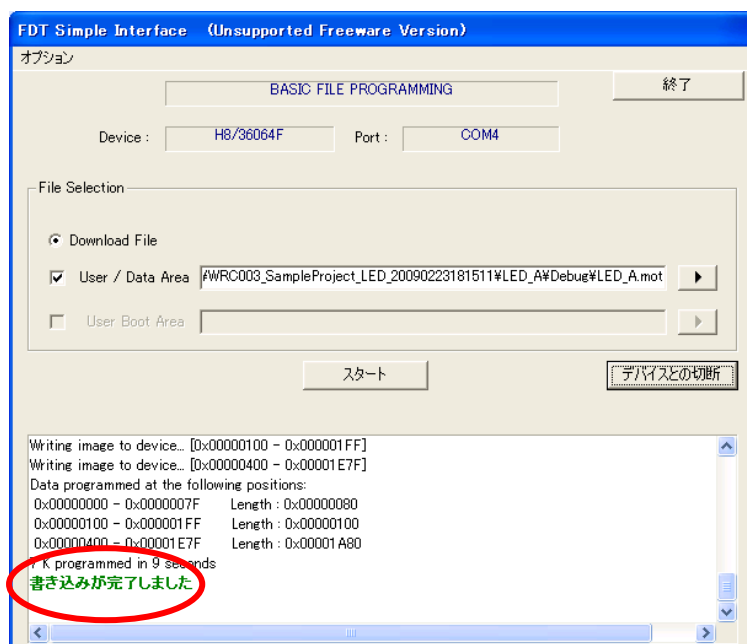



図 1-63　書き込み完了

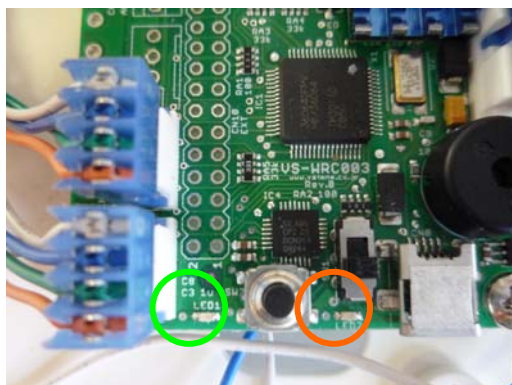
手順⑨　CPU ボードの電源を入れると，書き込んだプログラムが実行されます．正常に書き込めている場合，CPU 上のオレンジと緑の LED が交互に点滅します．

LED（左：緑）　LED（右：オレンジ）

図 1-64　CPU ボード上の LED

なお，設定は次回起動時も保存されていますので，同時の手順のみで書き込むことができます．

2. C 言語プログラミングで実際にロボットを動かしてみよう

2.1. LED を光らせよう

いよいよ実際に CPU ボードに記載されている機能を C 言語で動かしてみます. みなさん, LED とはどのようなものかご存知ですか？LED とは発光ダイオードのことで, 電圧をかけ電流を流すことができます. VS_WRC003 には緑とオレンジの２つの LED を搭載しています.

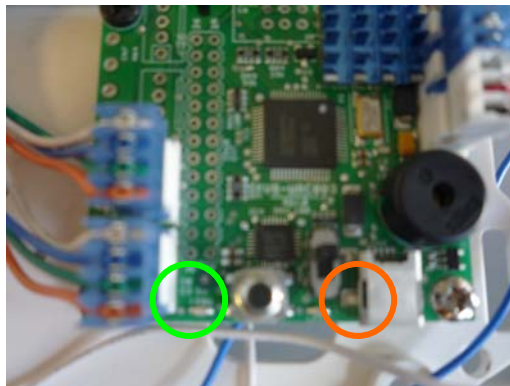



図 2-1　VS-WRC003 上に搭載している LED

CPU ボードの LED を点灯, 消灯させるには, 関数 LED （), 関数 Wait （) などを使って制御します. これらの関数はヘッダファイル VS_WRC003.h 内で定義され, プログラム内で使用することができます. ここで使用する関数を以下で説明します.

○ void LED（BYTE LedOn）;

LED () 関数は VS_WRC003 内にある２つの LED の, 点灯, 消灯を引数（※） [LedOn] に数値を与えることで制御することができます.

(※) 関数に渡したい情報があるときは, 関数名に続く （) に宣言します. これを引数といいます.

[LedOn] に与える数値は 2 進数で, bit0 が緑, bit1 がオレンジ, 1 のとき全点灯, 0 のとき全消灯します. 引数は以下で説明します.

値 (10 進数)	LED (オレンジ)	LED (緑)
0	消灯	消灯
1	消灯	点灯
2	点灯	消灯
3	点灯	点灯

表 1.1 指定した値と LED の状態

[引数]

0：消灯　1：緑　2：オレンジ　3：両方
[戻り値]
無し

例：LED（0）；　　　　　　　　//LED 消灯
　　LED（1）；　　　　　　　　//LED 緑色点灯
　　LED（2）；　　　　　　　　//LED オレンジ色点灯
　　LED（3）；　　　　　　　　//LED 全点灯

○ void Wait（int msec）；

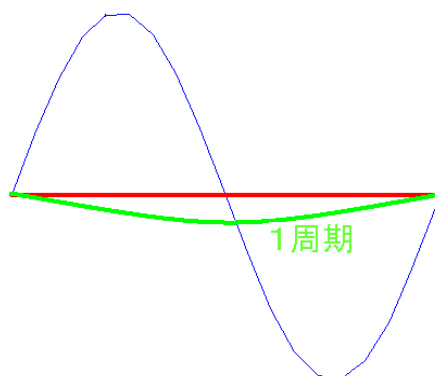
Wait（）関数は，引数［msec］で与えた時間（単位は msec ）だけ待つ関数です．Msec とは，ミリ秒のことで 1,000 分の 1 秒のことです．例えば Wait（1000）は 1 秒（1=1000msec）だけ待つ処理になります．

[引数]
待ち時間（単位：msec）
[戻り値]
無し

例：Wait（2000）；　　　　　　// 2 秒待つ
　　Wait（3500）；　　　　　　// 3.5 秒待つ
　　Wait（10000）；　　　　　　//１０秒待つ

周期　周波数

周期…定期的に同じことが繰り返される処理において，はじめの状態に戻るまでの時間のことを周期といいます．

[例]

周波数…周期の逆数で表され，1 秒間に何回振動するかを表しています．単位は Hz で電波などにも用いられています．

○ void Sync（）；

Sync（）は同期用の関数で，実行したとき，Init（）関数で設定した周波数で同期し、処理を戻す関数です.

例：Maincycle を 60Hz に設定した場合，以下のソースでは，処理 A が 60Hz で実行される.

ただし，処理 A が 1/60sec 以上かかる場合は意味を成さない.

```
while(1){
        Sync();
        処理 A;
}
```

[引数]

無し

[戻り値]

内部で待ったカウント（単位無し）．０の場合，設定した周波数で同期できていない.

MainCycle を 50(Hｚ)に設定したとき(１回のループは 0.02sec)

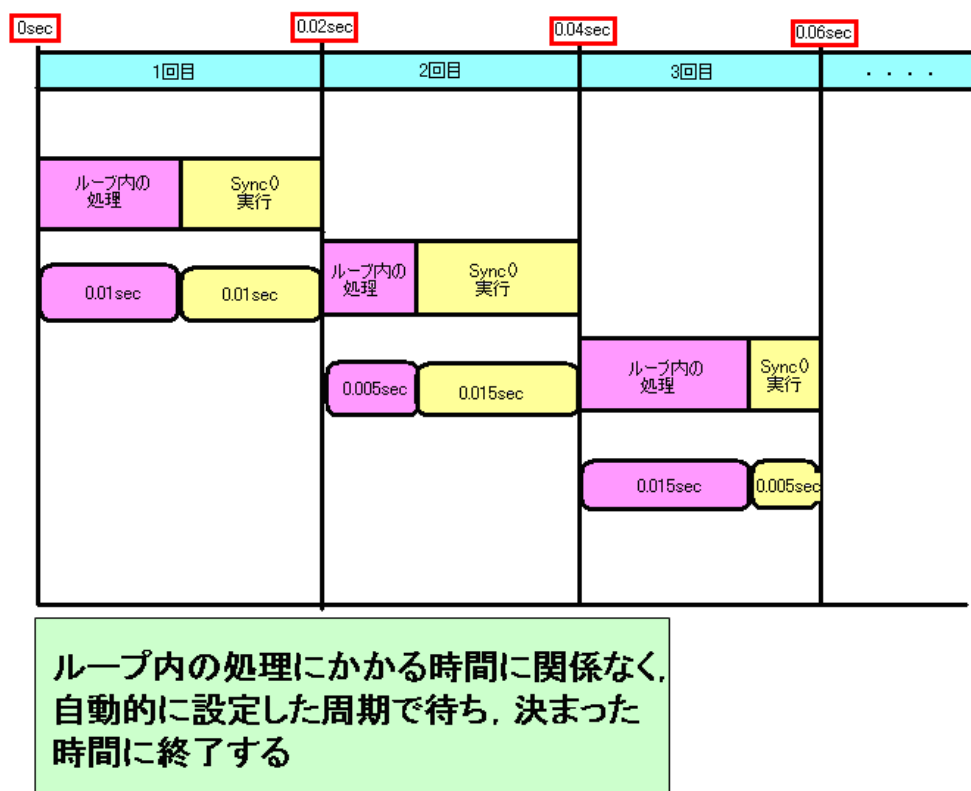



図 2-2　関数 Sync を使った場合のプログラム処理

○ void Init（BYTE MainCycle）；

　CPU 全体の初期化関数．Sync（）関数で利用する周波数を与える．

[引数]

Sync 関数の同期周波数（単位は Hz、30Hz 以上）．

[戻り値]

無し

○ BYTE getMainCycle（）；

　Sync 関数の同期周波数取得．Init で設定した同期周波数を取得する．

[引数]

無し

[戻り値]

同期周波数（Hz）

2.1.1.　実際に LED を光らせてみよう

　では，実際に LED を光らせるプログラムを書いてみましょう！

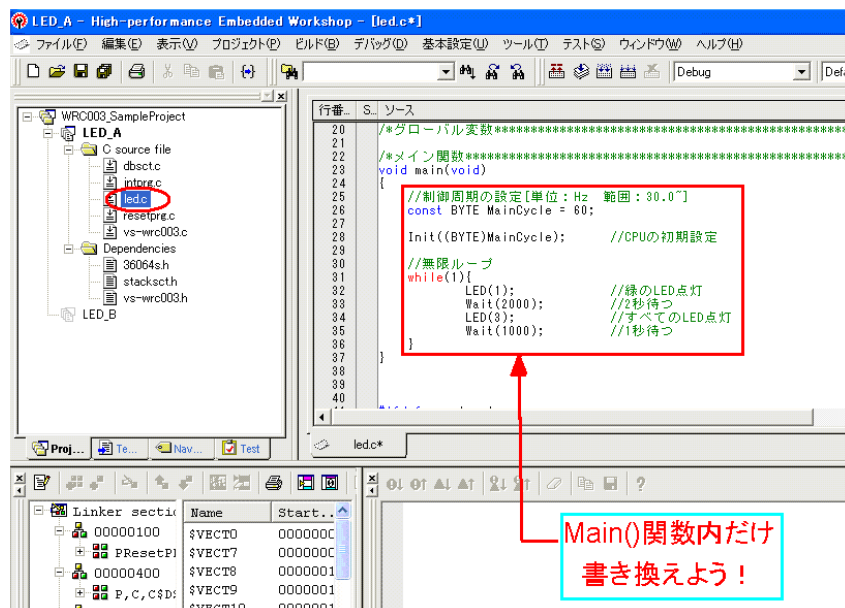
みなさんが書き換えるのは led.c 内の main 関数です．

[注意！！] led.c 以外の他のフォルダは書き換えないで下さい．

プログラムが正常に動作しない可能性があります．

使用できないライブラリ関数が存在します．

[例]printf 関数，scanf 関数など

図 2-3　関数内の書き換え場所

では実際に LED が光るプログラムを led.c の main() 関数内に書き込んでみましょう！LED のプログラムを以下に示します.

```
[1] void main(void)
[2] {
[3]         //制御周期の設定[単位：Hz　範囲：30.0～]
[4]         const BYTE MainCycle=60;
[5]         Init((BYTE)MainCycle);    //CPU の初期設定
[6]         //無限ループ
[7]         while(1){
[8]                 LED(2);           //オレンジの LED 点灯
[9]                 Wait(500);        //0.5 秒待つ
[10]                LED(3);           //全 LED 点灯
[11]                Wait(1000);       //1 秒待つ
[12]                }
[13]  }
```

[プログラム解説]

- 1 行目は関数の宣言です.
- 4 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています. また，MainCycle 関数により Sync 関数を使う周波数を 60 取得.
- 5 行目は Init 関数により，CPU 全体の初期化を行っています. また，キャストを行うことにより型を BYTE に一時的に変更しています.
- 7 行目はメインループで，while 文で書かれた無限ループ内に実行したい処理を記述します.
- 8 行目は LED がオレンジ色に点灯します.
- 9 行目は 0.5 秒待ちます. Wait（）関数を実行することでオレンジ色が 0.5 秒点灯することが出来ます.
- 10 行目は全ての LED が点灯します.
- 11 行目は 1 秒待ちます.

☆　フローチャート☆

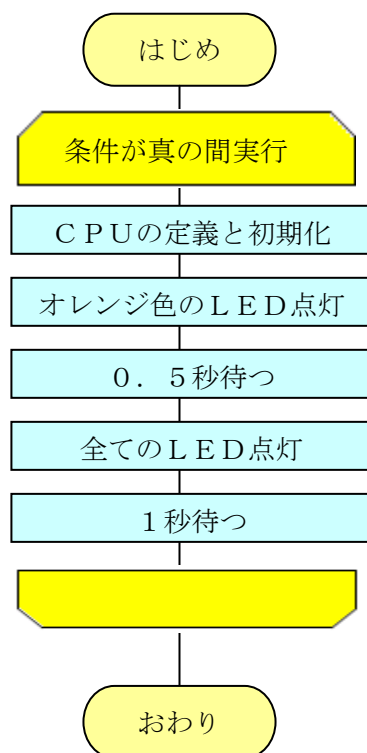



図 2-4

例題 1

LED（０）～LED（４）まで，引数の値によって LED がどのように変化するのか確認し，自分で点灯の間隔も好きなように調節してみよう．

[解答例]

```
[1] void main(void)
[2]   {
[3]               //制御周期の設定[単位：Hz　範囲：30.0～]
[4]               const BYTE MainCycle=60;
[5]               Init((BYTE)MainCycle);      //CPU の初期設定
[6]               //無限ループ
[7]               while(1){
[8]                           LED(1);           //緑色の LED 点灯
[9]                           Wait(2000);       // 2 秒待つ
[10]                          LED(2);           //オレンジ色の点灯
[11]                          Wait(1000);       //1 秒待つ
[12]                          LED(0);           //LED 消灯
[13]                          Wait(1500);       //1.5 秒待つ
[14]                          LED(3);           //すべての LED 点灯
[15]                          Wait(600);        //0.6 秒待つ
[16]                         }
[17]      }
```

[プログラム解説]

・1 行目は関数の宣言です.

・4 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています．また，MainCycle 関数により Sync 関数を使う周波数を 60 取得.

・5 行目は Init 関数により，CPU 全体の初期化を行っています．また，キャストを行うことにより型を BYTE に一時的に変更しています.

・7 行目はメインループです.

・8 行目は LED が緑色に点灯します.

・9 行目は 2 秒待ちます.

・10 行目はオレンジ色の LED が点灯します.

・11 行目は 1 秒待ちます.

・12 行目は LED が消灯します.

・13 行目は 1.5 秒待ちます.

・14 行目は全ての LED が点灯します.

・15 行目は 0.6 秒待ちます.

☆フローチャート☆

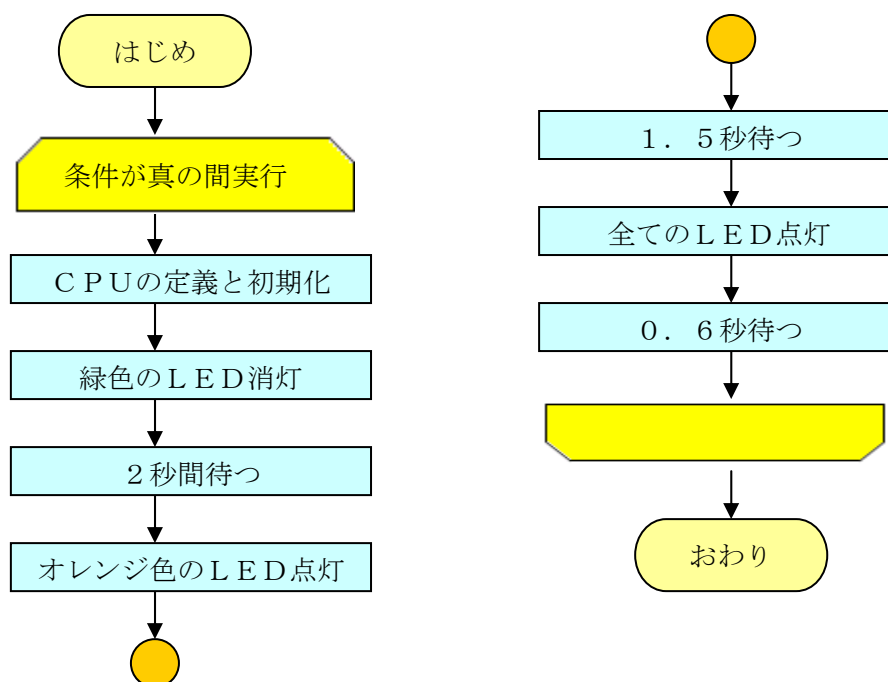



図 2-5

○ BYTE getSW（）

ボタン状態取得を表します．CPU ボード上の押しボタンの状態を取得するときに用います．

[引数]

無し

[戻り値]

0:ボタンが押されていない　1:ボタンが押されている

例題 2

スイッチを押したら緑色とオレンジ色の LED が交互に点灯するプログラムを作成せよ．

[解答例]

```
[1]void main(void)
[2]{
[3]             //制御周期の設定[単位：Hz　範囲：30.0～]
[4]             const BYTE MainCycle=60;
[5]             Init((BYTE)MainCycle);        //CPU の初期設定
[6]             //無限ループ
[7]             while(1){
[8]                     if(getSW()==1){        //スイッチが押されていたら
[9]                       LED(1);              //緑色の LED 点灯
[10]                      Wait(1000);          // 1 秒待つ
[11]                     LED(2);               //オレンジ色の点灯
[12]                     Wait(1000);           //1 秒待つ
[13]                    }
[14]                 else{
[15]                     LED(0);               //LED 消灯
[16]                   }
[17]                 }
[18] }
```

[プログラム解説]

- 1 行目は関数の宣言です．
- 4 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています．また，MainCycle 関数により Sync 関数を使う周波数を 60 取得．
- 5 行目は Init 関数により，CPU 全体の初期化を行っています．また，キャストを行うことにより型を BYTE に一時的に変更しています．
- 7 行目はメインループです．
- 8～16 行目は if 文によりスイッチが押されているかどうか判定します．もし押されていたら if 文内のループが実行され 1 秒間緑色の LED が点灯し，1 秒間オレンジ色の LED が点灯します．スイッチが押されなかっかったら LED が消灯します

☆　フローチャート☆

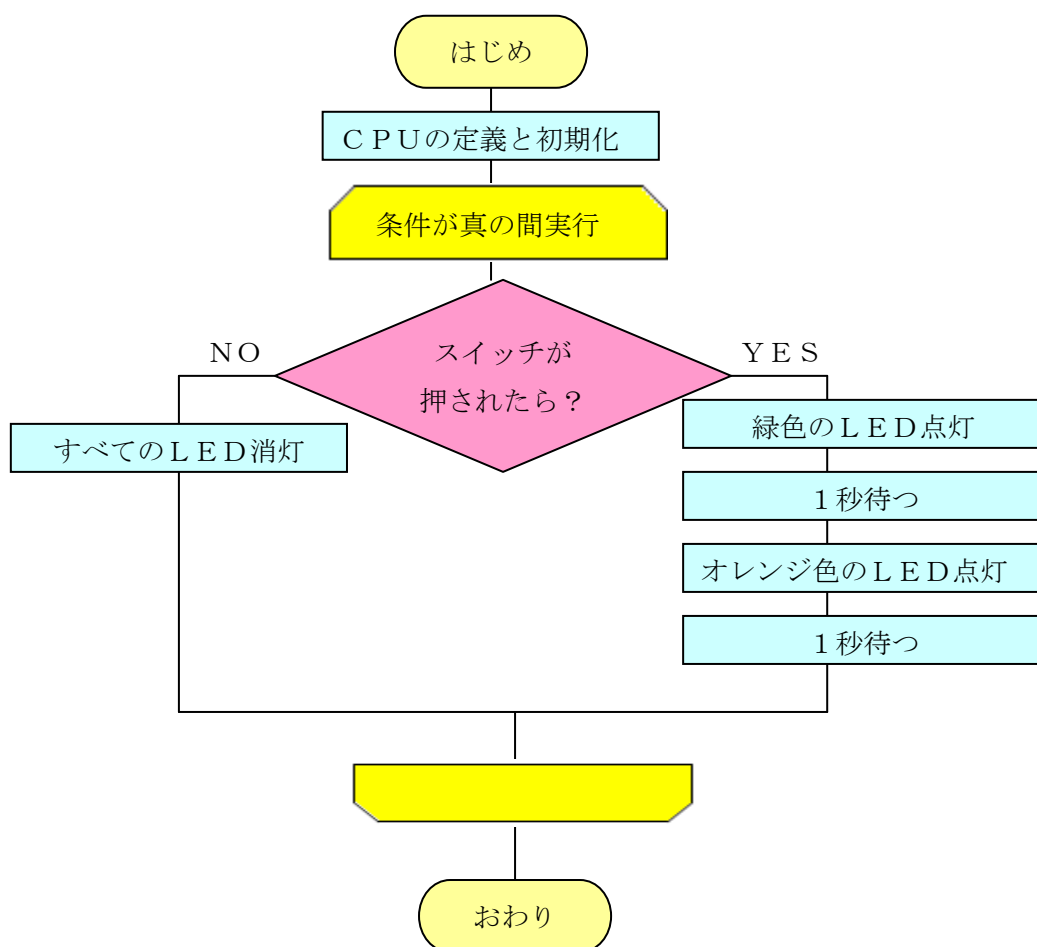



図 2-6

応用問題

最初の状態は全ての LED が点灯し，スイッチを押したらオレンジ色の LED が点灯するプログラムを作成し，フローチャートの作成も行おう．

[解答例]

```
[1]void main(void)
[2]{
[3]        //制御周期の設定[単位：Hz　範囲：30.0~]
[4]        const BYTE MainCycle = 60;
[5]        Init((BYTE)MainCycle);            //CPU の初期設定
[6]        LED(3);                           //全ての LED 点灯
[7]        while(1){                         //メインループ
[8]                if(getSW()){              //スイッチが押されたら
[9]                        LED(2);           //オレンジ色のの LED 点灯
[10]                       }
[11]               }
[12]}
```

[プログラム解説]

- 1 行目は関数の宣言です.
- 4 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています. また，MainCycle 関数により Sync 関数を使う周波数を 60 取得.
- 5 行目は Init 関数により，CPU 全体の初期化を行っています. また，キャストを行うことにより型を BYTE に一時的に変更しています.
- 6 行目はすべての LED が点灯します.
- 7 行目はメインループです.
- 8～9 行目は if 文によりスイッチが押されているかどうか判定します. もし押されていたら if 文内のループが実行され，オレンジ色のＬＥＤが点灯します.

☆フローチャート☆

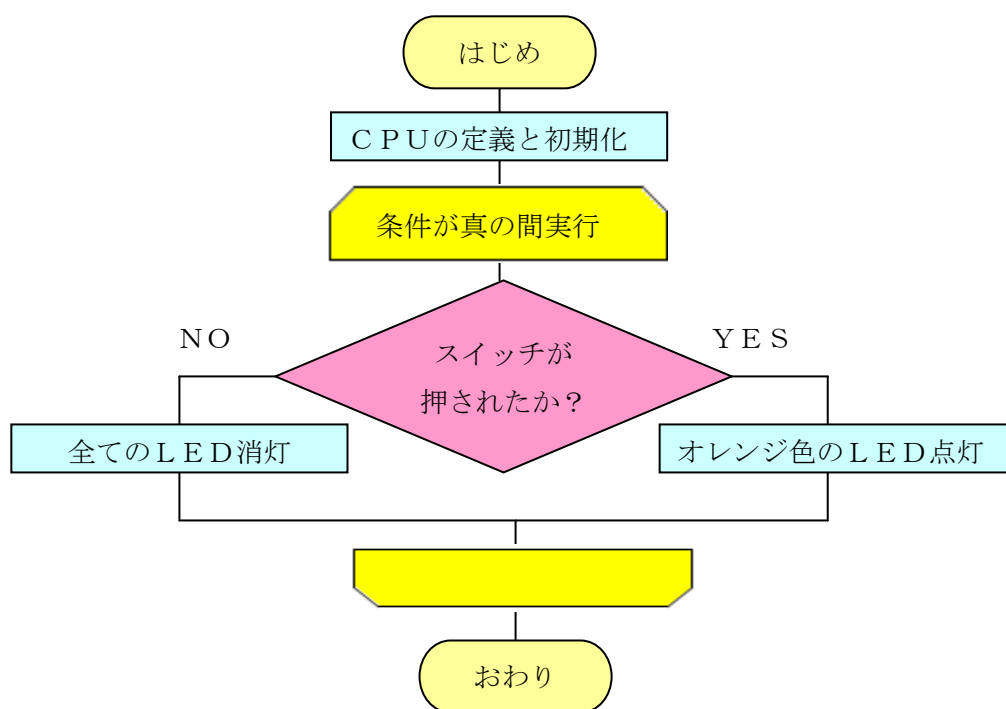



図 2-7

2.2. 10 進数，2 進数，16 進数

～10 進数～

10 進数は，0 から 9 までの 10 個の数字を使って数を表現します.

数は，0，1，2，3，4，5，6，7，8，9 と順に増え，次に位が増えて 10 になります.
このようにして，10 進数は，1，10，100，1000，10000…と位が繰り上がります，
1 は 10 の 0 乗（10^0），10 は 10 の 1 乗（10^1），100 は 10 の 2 乗（10^2），1000 は 10 の 3 乗（10^2）
…と言い換えることができます.
ですから，10 進数は，10^0，10^1，10^2，10^2…と位が繰り上がるとも言えます.
　例えば 10 進数で 2976 という数は，以下のように表すことができます.

10^3の位	10^2の位	10^1の位	10^0の位
2	9	7	6

図 2-8　10 進数の例

これを，数式で以下のように表すことができます.

$$2\times10^3 + 9\times10^2 + 7\times10^1 + 6\times10^0 = 2\times1000 + 9\times100 + 7\times10 + 6\times1 = 2976$$

～2 進数～

　2 進数は，数字 0，1 の 2 個の数字を使って数を表現します.
数は，0，1 と順に増え，次に位が増えて 10 になります.
このようにして，2 進数は，2^0（1），2^1（2），2^2（4），2^3（8）…と位が繰り上がります.（( )
内は 10 進数での数)
　例えば 2 進数で 1101 という数は，以下のように表すことができます.

2^3の位	2^2の位	2^1の位	2^0の位
1	1	0	1

図 2-9　2 進数の例

これを，数式で以下のように表すことができます.

$$1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 1\times8 + 1\times4 + 0\times2 + 1\times1 = 13$$ （10 進法）

　逆に，10 進数から 2 進数へ変換するには，10 進数を 2 で割って，その商をさらに 2 で割

る，またその商を 2 で割って…と，余りを出しながら商が 0 になるまで繰り返します．そして最後の余りを先頭に下から順に並べます．

例えば 10 進数で 19 という数は，以下のように計算することができます．

図 2-10　2 進数を求める計算（10 進法）

10進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2進数	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000

図 2-11　2 進数，10 進数対応表

～16 進数～

16 進数は 0 から 9 までの数字と A から F までのアルファベットを使って数を表現します．数は，0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F と順に増え，次に位が増えて 10 になります．

A は 10 進数で 10，B は 10 進数で 11，C は 10 進数で 12，D は 10 進数で 13，E は 10 進数で 14、F は 10 進数で 15 です．

このようにして、16 進数は 16^0（1）、16^1（16）、16^2（256）、16^3（4096）…と位が繰り上がります。(( ) 内は 10 進数での数)

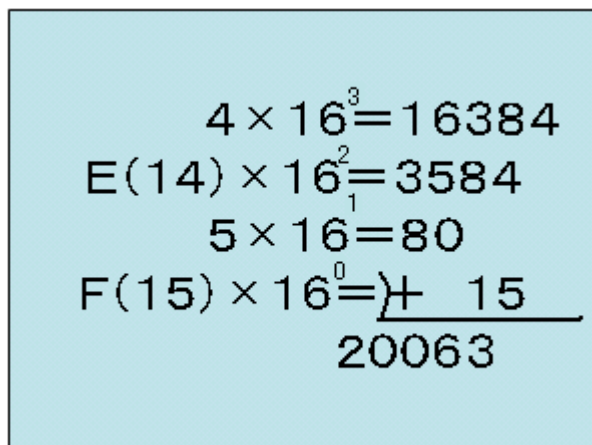
例えば 4E5F という 16 進数は、以下のように表せます．

16^3の位	16^2の位	16^1の位	16^0の位
4	E	5	F

図 2-12　16 進数の例

これを，数式で以下のように表すことができます.

4×16^3 ＋ $E\times16^2$ ＋ 5×16^1 ＋ $F\times16^0$ ＝ 4×4096 ＋ E（14）×256 ＋ 5×16 ＋ F（15）×1 ＝ 20063（10 進法）

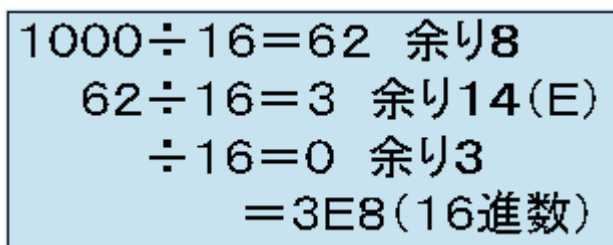
実際に 16 進数を 10 進数に変えるときは，以下のように縦算にすると簡単です.

$4\times16^3=16384$
$E(14)\times16^2=3584$
$5\times16^1=80$
$F(15)\times16^0=1 + 15$
20063

図 2-13　16 進数の例

逆に，10 進数から 16 進数へ変換するには，10 進数を 16 で割って，その商をさらに 16 で割る，またその商を 16 で割って…と，余りを出しながら商が 0 になるまで繰り返します.そして最後の余りを先頭に下から順に並べます.

例えば 10 進数で 1000 という数は、以下のように計算することができます.

1000÷16＝62　余り8
62÷16＝3　余り14（E）
÷16＝0　余り3
＝3E8（16進数）

図 2-14　2 進数，10 進数対応表

ブザーを用いた問題でこれらの計算法を使う問題が出ますので覚えておきましょう.

2.3. ブザーを鳴らしてみよう

次に同じく CPU ボードに搭載されているブザーもプログラムで鳴らしてみましょう.

CPU ボード上には圧電ブザーが搭載されています. 圧電ブザーとは, 電圧を印加することにより形状を歪ませせることにできる圧電素子を用いたブザーです. 圧電素子とは電圧を印加すると歪み, 印加しないと元に戻ります. このときの振動を利用してブザーを鳴らすことが出来ます.

電圧ブザーには, 発振回路が内蔵されていて電圧を印加するだけで鳴らすことのできる自動式と, 発振回路を内蔵していない他励式に分けられ, VS_WRC003 には他励式が搭載されています. 他励式のブザーは鳴らしたい周波数でオン/オフすることで音程を変更することができ, オン/オフのデューティー比（※）を変更することでボリュームを若干変更することが出来ます. VS_WRC003 では, ブザーが直接 CPU ボードに接続されているため, プログラムから音を出すことも出来ます.

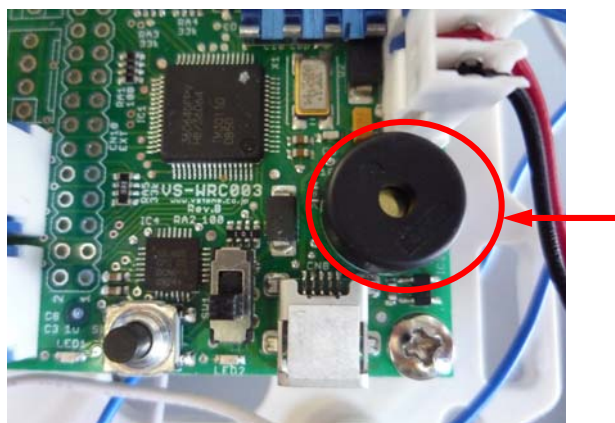



図 2-15 VS-WRC003 上に搭載している LED

（※）デューティー比

マイクロコンピュータからブザーやモータなどを出力されるパルス（電圧の High・Low）の比率です.

ブザーの制御には関数 BuzzerSet（）, BuzzerStart（）, BuzzerStop（）を利用します. 各関数は以下で解説します.

○ void BuzzerSet（BYTE pitch, BYTE vol）

ブザーの音程（引数 pitch）, ボリューム（引数 vol）を設定する関数です. どちらも Byte データ（0～255）までで指定します. 音程は数値を小さくすると高くなり, 大きくすると低い音になります. この関数をブザーが鳴っているときに実行すると, ブザーを自動的に停止し値を設定します.

[引数]

pitch：音程の設定（0～255, 値が大きいほど低い音）

vol：ボリュームの設定（0～128）

[戻り値]

無し

音階		数値
低		
B	シ	２３３
C	ド	２２０
D♭		２０８
D	レ	１９６
E♭		１８５
E	ミ	１７５
F	ファ	１６５
G♭		１５６
G	ソ	１４７
A♭		１３９
A	ラ	１３１
B	シ	１１７
C	ド	１１０
高		

表 2.1 音階と数値の対応

○ void BuzzerStart（）

ブザーを鳴らし始めます．BuzzerStop（）関数を呼ぶまで鳴り続けます.

[引数]

無し

[戻り値]

無し

○ void BuzzerStop（）

鳴っているブザーを止めます.

[引数]

無し

[戻り値]

無し

上記の三つの関数を使ったプログラムの main() 関数を以下に示します.

例題

スイッチを押す度にブザーの音程が高くなるプログラムを作成せよ.

[解答例]

```
[1]void main(void)
[2]{
[3]	//制御周期の設定[単位：Hz　範囲：30.0~]
[4]	const BYTE MainCycle = 60;
[5]	 Init((BYTE)MainCycle);		//CPU の初期設定
[6]	BYTE pitch = 18;		//音程の設定用変数の宣言
[7]	BuzzerSet(0x80,0x80);		//ブザーの設定
[8]	//ループ
[9]	while(1){
[10]		if(getSW()){
[11]			pitch　−= 18;		//音程を高くする
[12]			BuzzerSet(pitch,0x80);	//音程の設定
[13]			BuzzerStart();		//ブザーを鳴らし始める
[14]			Wait(2000);		//2 秒待つ
[15]			BuzzerStop();		//ブザーを止める
[16]		}
[17]	}
[18]　}
```

[プログラム解説]

- 1 行目は関数の宣言です.
- 4 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています.また，MainCycle 関数により Sync 関数を使う周波数を 60 取得.
- 5 行目は Init 関数により，CPU 全体の初期化を行っています．また，キャストを行うことにより型を BYTE に一時的に変更しています.
- 6 行目は変数 pitch の宣言と初期化を行っています.
- 7 行目はブザーの設定をしています.
- 9～17 行目はメインループで，スイッチの値を取得し，戻り値が 1 だったら pitch に 18 を減算しブザーを 2 秒間鳴らすようにしています．音程は減算すると上がりますので，押されるごとに 18 を減算するプログラムになっています.

POINT

BuzzerSet(0x80,0x80);の 0x80 はどういう意味？と思った人も多いかもしれません.

0x80 は実は 16 進数で表されています．16 進数を数字で書くときは先頭に「**0x**」を付けて書きます．16 進数を 10 進数にする方法は，まず 2 進数に変換してから 10 進数に変換します.

手順①　0x80（16 進数）→10000000（2 進数）に変換.

８÷２＝４余り０
４÷２＝２余り０
２÷２＝１余り０
１÷２＝０余り１
＝１０００（２進数）

手順②　10000000（2 進数）→128（10 進数）に変換．

$1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0 = 1\times128 =$ 128（10 進法）

つまり，16 進数で表されている「0x80」は，10 進数に変換すると「128」になります

☆　フローチャート☆

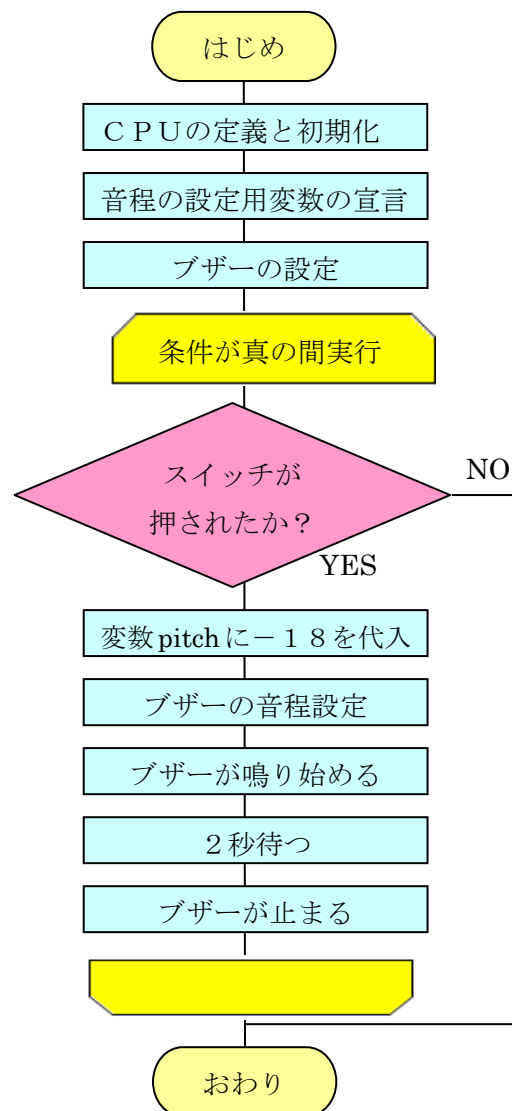



図 2-16

応用問題

表 2.1 を参考にしてボタンを押したら, かえるの歌がブザーから流れるプログラムを作成せよ.

[解答例]

```
[1]/*関数の宣言*************************************************************/
[2]void enso(char oto[],int msec);          //ブザーの音程を変化させる関数
[3]
[4]void main(void)
[5]{
[6]        int i,j,k,m;                      //カウント用変数
[7]        //制御周期の設定[単位：Hz　範囲：30.0~]
[8]        const BYTE MainCycle = 60
[9]
[10]       Init((BYTE)MainCycle);            //CPU の初期設定
[11]       BuzzerSet(0x80,0x80);             //ブザーの設定
[12]       LED(2);                           //オレンジ色の LED 点灯
[13]
[14]       //ループ
[15]        while(1){
[16]               if(getSW()){              //もしスイッチが押されたら
[17]       //ド
[18]                       enso("do",500);
[19]       //レ
[20]                       enso("re",500);
[21]       //ミ
[22]                       enso("mi",500);
[23]       //ファ
[24]                       enso("fa",500);
[25]       //ミ
[26]                       enso("mi",500);
[27]       //レ
[28]                       enso("re",500);
[29]       //ド
[30]                       enso("do",1000);
[31]       //ミ
[32]                       enso("mi",500);
[33]       //ファ
[34]                       enso("fa",500);
[35]       //ソ
[36]                       enso("so",500);
[37]       //ラ
[38]                       enso("ra",500);
[39]       //ソ
[40]                       enso("so",500);
[41]       //ファ
[42]                       enso("fa",500);
[43]       //ミ
[44]                       enso("mi",1000);
```

```
[45]
[46]
[47]                    for(i=0;i<4;i++){
[48]     //ド
[49]                    enso("do",850);
[50]
[51]                    BuzzerStop();
[52]                    Wait(100);
[53]
[54]              }
[55]              for(m=0;m<2;m++){
[56]     //ド
[57]                    enso("do",250);
[58]              }
[59]
[60]              for(j=0;j<2;j++){
[61]     //レ
[62]                    enso("re",250);
[63]              }
[64]              for(k=0;k<2;k++){
[65]     //ミ
[66]                    enso("mi",250);
[67]              }
[68]              for(j=0;j<2;j++){
[69]     //ファ
[70]                    enso("fa",250);
[71]              }
[72]     //ミ
[73]                    enso("mi",500);
[74]     //レ
[75]                    enso("re",500);
[76]     //ド
[77]                    enso("do",500);
[78]              }
[79]                    else{                   //もしスイッチが押されなかったら
[80]                    BuzzerStop();           //ブザーが止まる
[81]                    }
[82]     }
[83]}
[84]/***************************************************************************[
[85]                   関数名：音程や音の長さを調節する関数
[86]                   引数：oto　音を格納する配列
[87]                        msec  ミリ秒数を指定する
[88]***************************************************************************/
[89]void enso(char oto[],int msec){
[90]     int ontei[]={223, 220, 196, 175, 165, 147, 131, 117,110};//音程が入った配列
[91]     int i;                                                    //変数
[92]
[93]     if(oto=="do")
[94]     {
```

```
[95]            i = 1;
[96]      }else if(oto == "re"){
[97]            i = 2;
[98]      }else if(oto == "mi"){
[99]            i = 3;
[100]     }else if(oto == "fa"){
[101]           i = 4;
[102]     }else if(oto == "so"){
[103]           i=5;
[104]     }else if(oto == "ra"){
[105]            i=6;
[106]      }else if(oto == "shi"){
[107]          i=7;
[108] }
[109]     BuzzerSet(ontei[i],0x80);                    //ブザーの設定
[110]     BuzzerStart();                               //ブザーが鳴り始める
[111]     Wait(msec);                                  //msec 秒待つ
[112]     }
```

[プログラム解説]

・１～2 行目はブザーの音程を変化させる関数の宣言です.

・４行目は関数の宣言です.

・6 行目はカウント用変数を宣言しています.

・8 行目は変数 const を使うことにより，その変数の値が書き換えるようにしています．また，MainCycle 関数により Sync 関数を使う周波数を 60 取得.

・10 行目は Init 関数により，CPU 全体の初期化を行っています．また，キャストを行うことにより型を BYTE に一時的に変更しています.

・11 行目はブザーの設定をしています.

・12 行目はオレンジ色のＬＥＤが点灯します.

・15～83 行目はメインループ内でそれぞれの音程や音の長さにあわせて関数を呼び出しています.

・16 行目はスイッチが押されたかどうかを if 文で判定しています.

・89 行目は関数の定義をしています．関数名は音程や音の長さを調節する関数です，引数は oto が音を格納する配列で， msec がミリ秒数を指定することで音の長さを調節しています,

・90 行目は int 型の ontei という名前の配列に表 2.1 にならって音程の値を格納します.

・91 行目は音を判定するために指定した変数です.

・93～95 行目はもし oto という配列に"do" が代入されたら i に 1 を代入します.

・96～97 行目はもし oto という配列に"re" が代入されたら i に 2 を代入します.

・98～99 行目はもし oto という配列に"mi" が代入されたら i に 3 を代入します.

・100～101 行目はもし oto という配列に"fa" が代入されたら i に 4 を代入します.

・102～103 行目はもし oto という配列に"so" が代入されたら i に 5 を代入します.

・104～105 行目はもし oto という配列に"ra" が代入されたら i に 6 を代入します.

・106～108 行目はもし oto という配列に"shi" が代入されたら i に 7 を代入します.

・109 行目はブザーの設定をしています

・112 行目はブザーが鳴り始めます.

・113 行目は指定した関数の呼び出しで指定した引数の値が呼び出され，指定したミリ秒ブザーが鳴り始めます.

☆フローチャート☆

図 2-17

音程や音の長さを調節する関数
音程を格納する配列の宣言
変数 i の宣言
oto=="do"
YES
NO
i=1
oto=="re"
YES
NO
i=2
oto=="mi
YES
NO
i=3
oto=="fa"
YES
NO
i=4
oto=="so"
YES
NO
i=5
oto=="ra"
YES
NO
i=6
oto=="shi"
YES
NO
i=7
ブザーの設定
ブザーが鳴り始める
msec 秒待つ
おわり

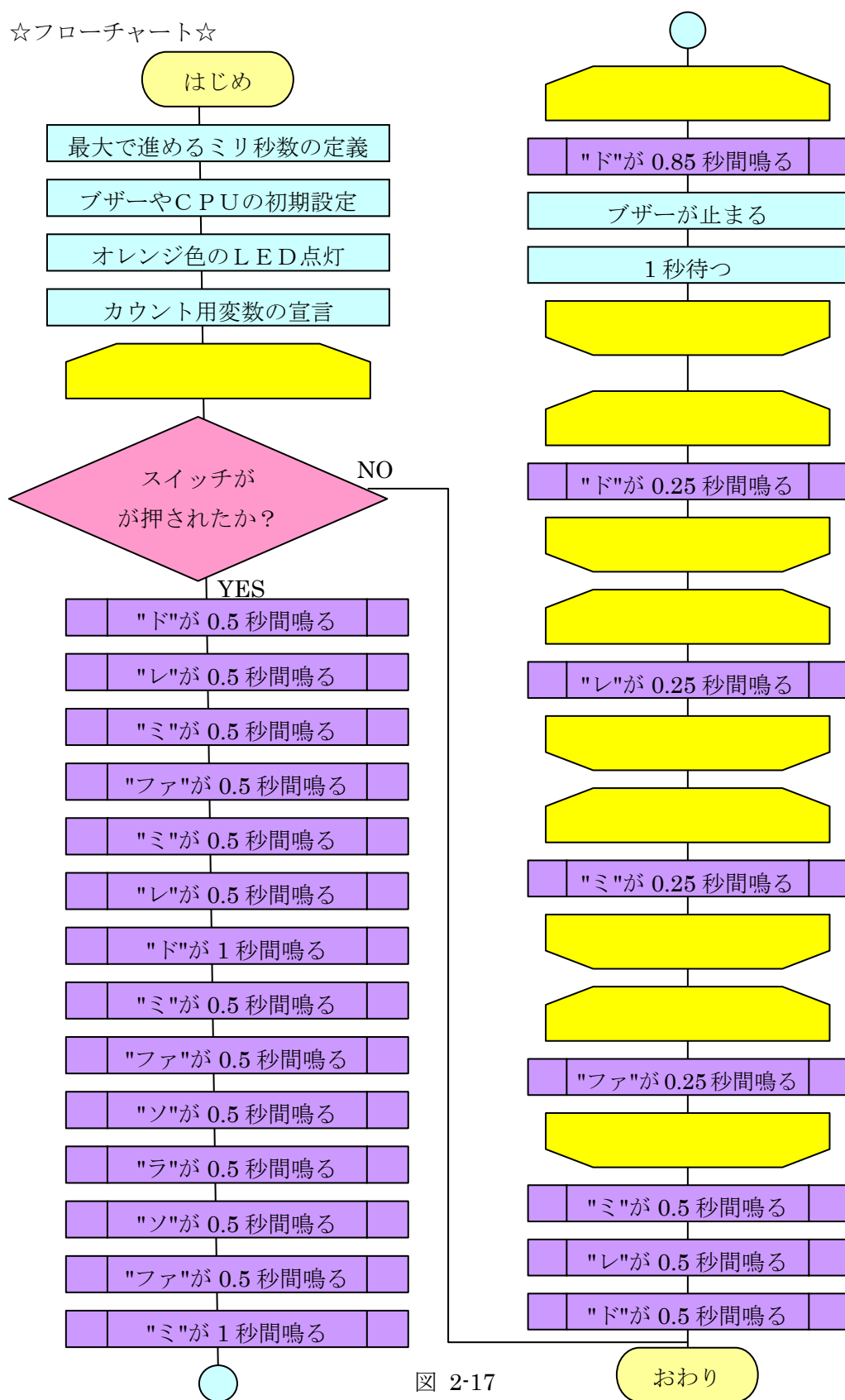
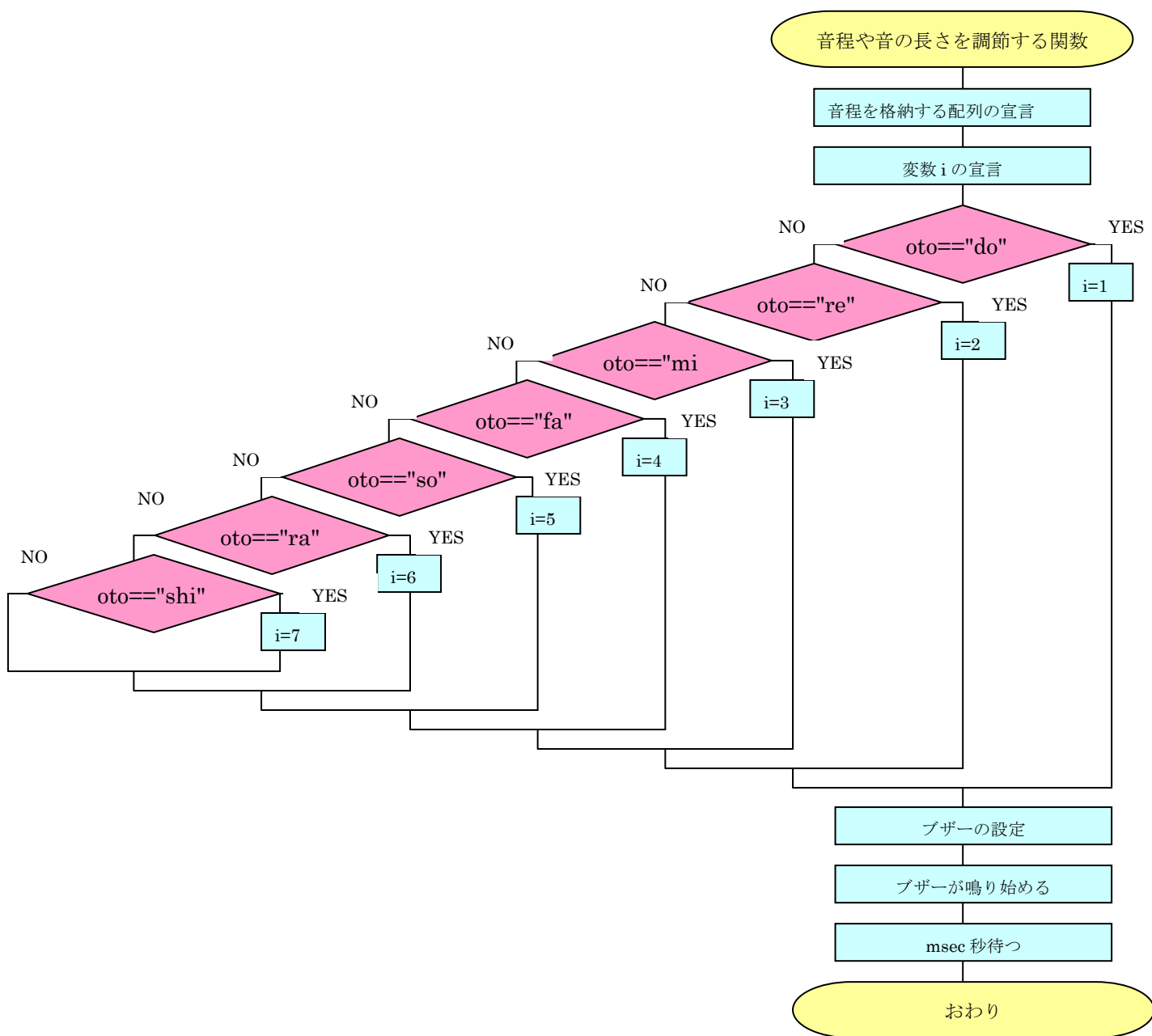



図 2-18

2.4. モータを制御してみよう

いよいよロボット製作にかかせないモータの制御です．関数を用いてモータを動かす基本的なプログラムを作成して，さまざまな例題でロボットが動くおもしろさを体験します．これで，一通りＣ言語でロボットを自由自在に動かすことが出来るようになります．

VS_WRC003 には 2ch の DC モータドライバが搭載されており，プログラムから二つのモータの回転方向や速度を制御することができます．搭載されているモータドライバにはＰWM端子があり，任意のデューティー比（※）の PMW を与えることで速度制御も可能になります．

※デューティー比：周期的なパルス波を出したときの周期とパルス幅の比のことです．

ＰWM制御

ＤＣ（直流電流）モータを印加する電圧を高くすれば速く，低くすれば遅く回転します．しかしマイコン（マイクロコンピュータ）から制御する場合，マイコンの出力は High（VS_WRC003 では 3.3V）を出力するか，Low を出力するかのどちらかなので，電圧を変えるためにはデジタルアナログ変換する外部回路が必要になります，そこで，PWM 制御を利用します．

PWM は，Piuse Width Modulation の略で，パルス（短時間の間に急峻な変化をする信号の総称）波のデューティー比を変化させて変調する変調方法です．

簡単に言うと，高速でモータのオン/オフを繰り返し，オンにする時間とオフにする時間の比を変えることで，あたかもアナログ的に電源電圧を可変して上げたかのような速度制御が可能になります．

図 2-19 の右の図はデューティー比が約 80%で，左の図はデューティー比が約 20%です．これを平均化すると，それぞれ High レベルの約 80%，約 20%の速度になります．数百Ｈｚ～数 KHｚの速い周期でパルスを与えることでスムーズな回転が得られます

．

デューティー比 20%　　　　デューティー比 80%

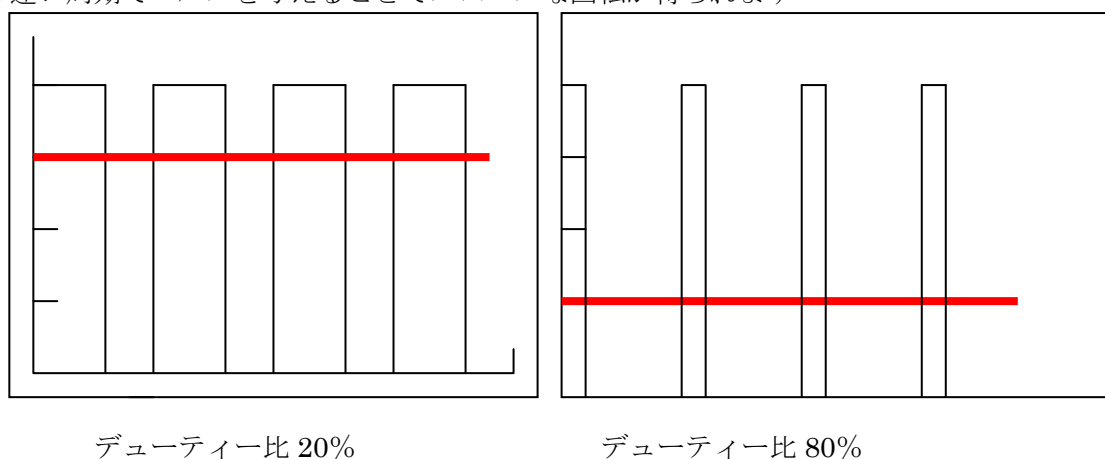



図 2-19

モータを制御するには関数 Mtr_Run（）を使用します．Mtr_Run（）について以下に説明します．

○void Mtr_Run（BYTE mt1, BYTE mt2, BYTE mt3, BYTE mt4）

Mtr_Run（）はモータの回転を設定する関数で，引数 mt1～mt4 に Byte データを与えることで，回転数，回転方向を設定します．（VS_WRC003 では mt3，mt4 は使用しません）

引数に 0 を与えることで停止，－128 を与えることでブレーキがかかります．また，正数を与えると正転，負数を与えると逆転します．

[引数]

BYTE mt1～mt4: モータの回転を設定．各値での動作については表 3.1 を参照してください．

正転	1～127　（0x01～0X7F）
逆転	－1～－127　（0xFF～0x81）
ブレーキ	－128　（0x80）
フリー	0　（0x00）

表 3.1 モータの動作

☆例☆

```
Mtr_Run(64,-64,0,0);     //モータの速さが高速で前進
Mtr_Run(-64,64,0,0);     //モータの速さが高速で後退
Mtr_Run(64,0,0,0);       //モータの速さが高速で左旋回
Mtr_Run(0,-64,0,0);      //モータの速さが高速で右旋回
```

Point

BeautoChaser のギアボックスは左右対称になっているため，前に走行するときにモータに与える回転は左右逆になります．前進は「Mtr_Run(64,64,0,0);」と一見間違えやすいですが，それでは BeautoChaser はどのような動きをするのでしょうか．プログラム中の引数を変えて試してみましょう．

[戻り値]

なし

上記の関数を使ったプログラムの main（）関数を以下に示します．まずは，以下の図のようにスイッチを押したら動くプログラムを例に示して説明します．

図 2-20

これから，Mtr_Run（）関数を使用する場合は，配布した資料の「Work Space」内の「サンプルプログラム　モータ制御」を使用しましょう！.

[解答例]

```
[1]#define MAX_TIME 5000                                   //最大に進めるミリ秒
[2]
[3]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御
[4]void R_turn_high(int sec,int x,double y,int i);
//速さが高速で右旋回し,走りすぎを制御
[5]/*メイン関数*****************************************************************/
[6]void main(void)
[7]{
[8]	int x;			//関数に実引数として変数 x を渡すための変数の宣言
[9]	double y;		//関数に実引数として変数 y を渡すための変数の宣言
[10]	int i;			//ループ用カウンタ変数
[11]
[12]	int F_high=12000;	//速さが高速で前進するミリ秒を指定する
[13]	int R_high=3300;	//速さが高速で右旋回するミリ秒を指定する
[14]
[15]	//制御周期の設定[単位：Hz　範囲：30.0~]
[16]	const BYTE MainCycle = 60;
[17]
[18]	Init((BYTE)MainCycle);		//CPU の初期設定
[19]	Mtr_Run(0,0,0,0);
[20]
[21]	//ループ
[22]	while(1){
[23]		if(getSW()){
[24]		/*高速の速さで前進し停止する*/
[25]		forward_high(F_high,x,y,i);			//関数の呼び出し
[26]
```

```
[27]                /*高速の速さで右旋回し停止する*/
[28]                R_turn_high(R_high,x,y,i);                          //関数の呼び出し
[29]
[30]                /*高速の速さで前進し停止する*/
[31]                forward_high(F_high,x,y,i);                         //関数の呼び出し
[32]                }
[33]        }
[34]}
[35]
[36]/**************************************************************************
[37]                        関数名：速さが高速で前進し,走りすぎを制御する関数
[38]                        引数:sec   進む時間(ミリ秒)
[39]                              x     割り算した結果の整数部分
[40]                              y     割り算した余り
[41]                              i     ループ用カウンタ変数
[42]************************************************************************** /
[43]void forward_high(int sec,int x,double y,int i)
[44]{
[45]        if(sec<=MAX_TIME){
[46]        Mtr_Run(64,-64,0,0);                        //高速で前進
[47]        Wait(sec);                                  //入力したミリ秒数だけ進む
[48]        Mtr_Run(0,0,0,0);                           //停止
[49]        LED(2);                                     //オレンジ色の LED 点灯
[50]        Wait(1000);                                 //1 秒間待つ
[51]        }
[52]        else{
[53]        x=sec/MAX_TIME;
[54]        y=sec%MAX_TIME;
[55]        for(i=0;i<x;i++){
[56]                Mtr_Run(64,-64,0,0);                //高速で前進
[57]                Wait(MAX_TIME);                     //MAX_TIME ミリ秒待つ
[58]                Mtr_Run(0,0,0,0);                   //停止
[59]                LED(1);                             //緑色の LED 点灯
[60]                Wait(1000);                         //1 秒間待つ
[61]                }
[62]        Mtr_Run(64,-64,0,0);                        //高速で前進
[63]        Wait(y);                                    //y ミリ秒待つ
[64]        Mtr_Run(0,0,0,0);                           //停止
[65]        LED(3);                                     //全 LED 点灯
[66]        Wait(1000);                                 //1 秒間待つ
[67]        }
[68]}
[69]
[70]/**************************************************************************
[71]                        関数名：速さが高速で右旋回し,走りすぎを制御する関数
[72]                              引数:sec   進む時間(ミリ秒)
[73]                                    x     割り算した結果の整数部分
[74]                                    y     割り算した余り
[75]                                    i     ループ用カウンタ変数
```

```
[76]************************************************************************ /
[77]void R_turn_high(int sec,int x,double y,int i)
[78]{
[79]	if(sec<=MAX_TIME){
[80]	Mtr_Run(0,-64,0,0);	//高速で右旋回
[81]	Wait(sec);	//入力したミリ秒数だけ進む
[82]	Mtr_Run(0,0,0,0);	//停止
[83]	LED(2);	//オレンジ色の LED 点灯
[84]	Wait(1000);	//1 秒間待つ
[85]	}
[86]	else{
[87]	x=sec/MAX_TIME;
[88]	y=sec%MAX_TIME;
[89]	for(i=0;i<x;i++){
[90]		Mtr_Run(0,-64,0,0);	//高速で右旋回
[91]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[92]		Mtr_Run(0,0,0,0);	//停止
[93]		LED(1);	//緑色の LED 点灯
[94]		Wait(1000);	//1 秒間待つ
[95]		}
[96]	Mtr_Run(0,-64,0,0);	//高速で右旋回
[97]	Wait(y);	//y ミリ秒待つ
[98]	Mtr_Run(0,0,0,0);	//停止
[99]	LED(3);	//全 LED 点灯
[100]	Wait(1000);	//1 秒間待つ
[101]	}
[102]}
```

[プログラム解答]

・1 行目は最大で進めるミリ秒数を定義しています．これはロボットもモータが長い間作動してしまいますと，ＣＰＵが熱くなってしまいＣＰＵが故障する可能性がでてきます．それを防ぐために最大で５秒まで作動すると定義しておきます．

・3〜4 行目は関数の宣言をしています．今回使用する関数は，速さが高速で前進し，走りすぎを制御する関数と速さが高速で右旋回し，走りすぎを制御する関数です．

・6 行目は関数の宣言です．

・8 行目はループ用カウンタの変数を宣言しています．

・9 行目は関数に実引数として変数 x を渡すための int 型の変数の宣言をしています．

・10 行目は関数に実引数として変数 y を渡すための double 型の変数の宣言をしています．

・12 行目は速さが高速で前進するミリ秒数を指定します．この値は forward_high 関数の引数の値として渡されます．

・13 行目は速さが高速で右旋回するミリ秒数を指定します．この値は R_turn_high 関数の引数の値として渡されます．

・16〜18 行目はＣＰＵの初期設定を行っています．

・19 行目は停止しています．

・22～34 行目のメインループ内ではもしスイッチが押されたら，高速で前進し停止する関数を呼び出し，高速で右旋回する関数を呼び出し，高速で前進し停止する関数を呼び出すというものです．

・36～42 行目は関数名と引数の説明を行っています．

・43 行目は関数の宣言を行っています．関数の型は void 型です．

・45～51 行目はもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，12 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです．

・52～58 行目まではもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 55 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します．この例を以下の[説明図]に表しました．

[説明図]

MAX_TIME の値が 5000

Sec の値が 7000 だとします

x=7000/5000=1

y=7000%5000=2000

中速で 5000 ミリ秒前進 →１秒停止している間ＬＥＤが緑色に点灯	For 文終了 →	中速で 2000 ミリ秒前進 →１秒停止している間全ての LED が点灯

・70～76 行目は関数名と引数の説明をしています．

・77 行目は関数の宣言を行っています．関数の型は void 型です．

・79～85 行目はもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，13 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです．

・86～102 行目まではもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 89 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します．

☆フローチャート☆

図 2-21

高速の速さで前進し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-22

高速の速さで右旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

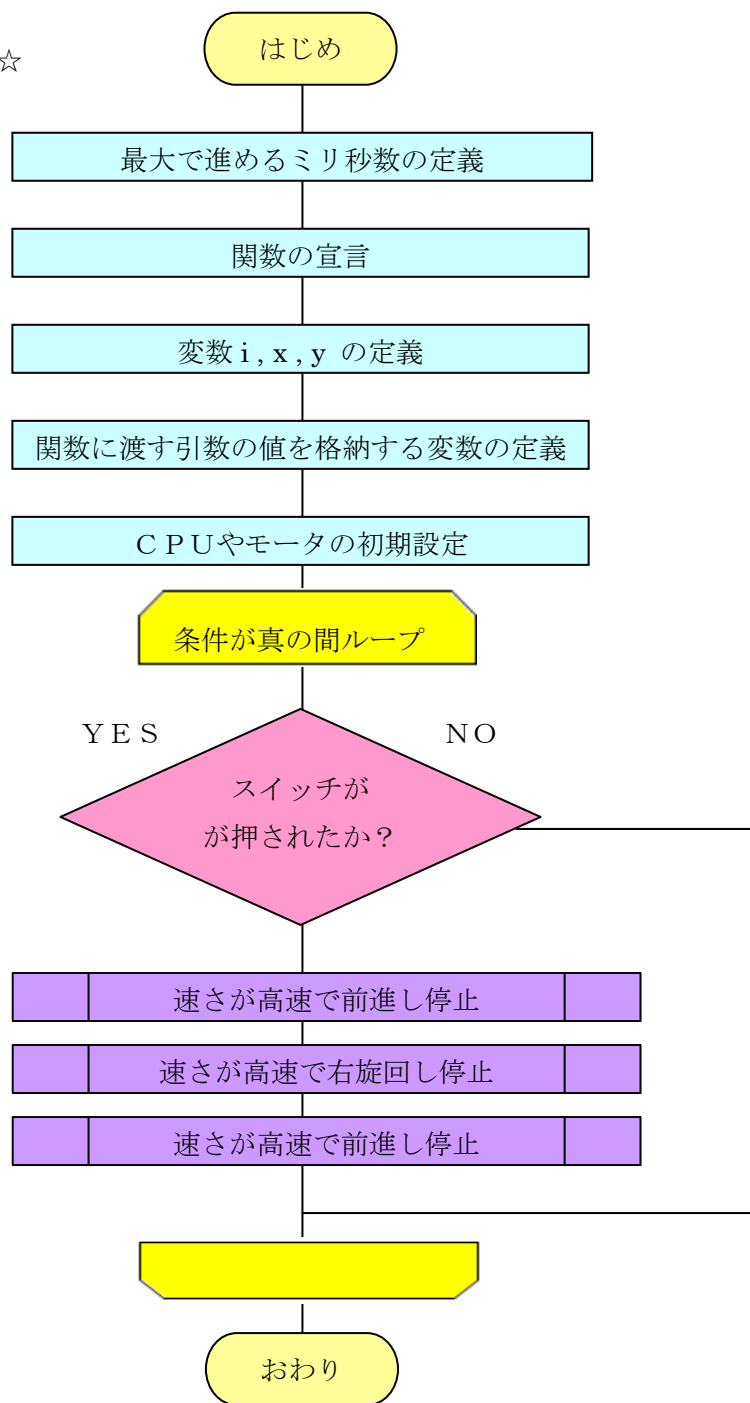



図 2-23

例題

配布した「サンプルプログラム　モータ制御」の一部を使用して中速で無限に三角形を書き続けるプログラムを作成しましょう.

[解答例]

```
[1]/*最大で進めるミリ秒数****************************************************/
[2]#define MAX_TIME 5000
[3]
[4]/*関数の宣言****************************************************************/
[5]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[6]void R_turn_high(int sec,int x,double y,int i);
//速さが高速で右旋回し,走りすぎを制御する関数
[7]
[8]/*メイン関数**************************************************************/
[9]void main(void)
[10]{
[11]	int i;						//ループ用カウンタ
[12]	int x;				//関数に実引数として変数 x を渡すための変数の宣言
[13]	double y;				//関数に実引数として変数 y を渡すための変数の宣言
[14]
[15]	int F_high=8000;		//速さが高速で前進するミリ秒を指定する
[16]	int R_high=4500;		//速さが高速で右旋回するミリ秒を指定する
[17]
[18]	//制御周期の設定[単位：Hz　範囲：30.0~]
[19]	const BYTE MainCycle = 60;
[20]	Init((BYTE)MainCycle);	//CPU の初期設定
[21]
[22]	//ループ
[23]	while(1){
[24]			/*高速の速さで前進し停止する*/
[25]			forward_high(F_high,x,y,i);			//関数の呼び出し
[26]
[27]			/*高速の速さで右旋回し停止する*/
[28]			R_turn_high(R_high,x,y,i);			//関数の呼び出し
[29]	}
[30]}
[31]
[32]/*************************************************************************
[33]			関数名：速さが高速で前進し,走りすぎを制御する関数
[34]			引数:sec　進む時間(ミリ秒)
[35]				x	割り算した結果の整数部分
[36]				y	割り算した余り
[37]				i	ループ用カウンタ変数
[38]************************************************************************ /
[39]void forward_high(int sec,int x,double y,int i)
[40]{
[41]	if(sec<=MAX_TIME){
[42]	Mtr_Run(64,-64,0,0);				//高速で前進
```

```
[43]        Wait(sec);                                  //入力したミリ秒数だけ進む
[44]        Mtr_Run(0,0,0,0);                           //停止
[45]        LED(2);                                     //オレンジ色の LED 点灯
[46]        Wait(1000);                                 //1 秒間待つ
[47]        }
[48]        else{
[49]        x=sec/MAX_TIME;
[50]        y=sec%MAX_TIME;
[51]        for(i=0;i<x;i++){
[52]                Mtr_Run(64,-64,0,0);                //高速で前進
[53]                Wait(MAX_TIME);                     //MAX_TIME ミリ秒待つ
[54]                Mtr_Run(0,0,0,0);                   //停止
[55]                LED(1);                             //緑色の LED 点灯
[56]                Wait(1000);                         //1 秒間待つ
[57]                }
[58]        Mtr_Run(64,-64,0,0);                        //高速で前進
[59]        Wait(y);                                    //y ミリ秒待つ
[60]        Mtr_Run(0,0,0,0);                           //停止
[61]        LED(3);                                     //全 LED 点灯
[62]        Wait(1000);                                 //1 秒間待つ
[63]        }
[64]}
[65]
[66]/******************************************************************************
[67]                関数名：速さが高速で右旋回し,走りすぎを制御する関数
[68]                        引数:sec    進む時間(ミリ秒)
[69]                             x      割り算した結果の整数部分
[70]                             y      割り算した余り
[71]                             i      ループ用カウンタ変数
[72]***************************************************************************** /
[73]void R_turn_high(int sec,int x,double y,int i)
[74]{
[75]        if(sec<=MAX_TIME){
[76]        Mtr_Run(0,-64,0,0);                                 //高速で右旋回
[77]        Wait(sec);                                          //入力したミリ秒数だけ進む
[78]        Mtr_Run(0,0,0,0);                                   //停止
[79]        LED(2);                                             //オレンジ色の LED 点灯
[80]        Wait(1000);                                         //1 秒間待つ
[81]        }
[82]        else{
[83]        x=sec/MAX_TIME;
[84]        y=sec%MAX_TIME;
[85]        for(i=0;i<x;i++){
[86]                Mtr_Run(0,-64,0,0);                           //高速でで右旋回
[87]                Wait(MAX_TIME);                               //MAX_TIME ミリ秒待つ
[88]                Mtr_Run(0,0,0,0);                             //停止
[89]                LED(1);                                       //緑色の LED 点灯
[90]                Wait(1000);                                   //1 秒間待つ
[91]                }
```

```
[92]      Mtr_Run(0,-64,0,0);                          //高速で右旋回
[93]      Wait(y);                                     //y ミリ秒待つ
[94]      Mtr_Run(0,0,0,0);                            //停止
[95]      LED(3);                                      //全 LED 点灯
[96]      Wait(1000);                                  //1 秒間待つ
[97]      }
[98]}
```

[プログラム解説]

・1～2 行目は最大で進めるミリ秒数を定義しています.

・4～6 行目は関数の宣言をしています．今回使用する関数は，速さが高速で前進し，走りすぎを制御する関数と速さが高速で右旋回し，走りすぎを制御する関数です.

・9 行目は関数の宣言です.

・11 行目はループ用カウンタの変数を宣言しています.

・12 行目は関数に実引数として変数 x を渡すための int 型の変数の宣言をしています.

・13 行目は関数に実引数として変数 y を渡すための double 型の変数の宣言をしています.

・15 行目は速さが高速で前進するミリ秒数を指定します．この値は forward_high 関数の引数の値として渡されます.

・16 行目は速さが高速で右旋回するミリ秒数を指定します．この値は R_turn_high 関数の引数の値として渡されます.

・18～20 行目はＣＰＵの初期設定を行っています.

・23～29 行目のメインループ内では高速で前進し停止する関数を呼び出し，高速で右旋回する関数を呼び出すというものです.

・32～38 行目は関数名と引数の説明を行っています.

・39 行目は関数の宣言を行っています．関数の型は void 型です.

・41～47 行目はもし 15 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，15 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.

・48～64 行目まではもし 15 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 51 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・66～72 行目は関数名と引数の説明をしています.

・73 行目は関数の宣言を行っています．関数の型は void 型です.

・75～81 行目はもし 16 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，16 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.

・85～97 行目まではもし 16 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME

を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 85 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します．

☆フローチャート☆

図 2-24

高速の速さで前進し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-25

高速の速さで右旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

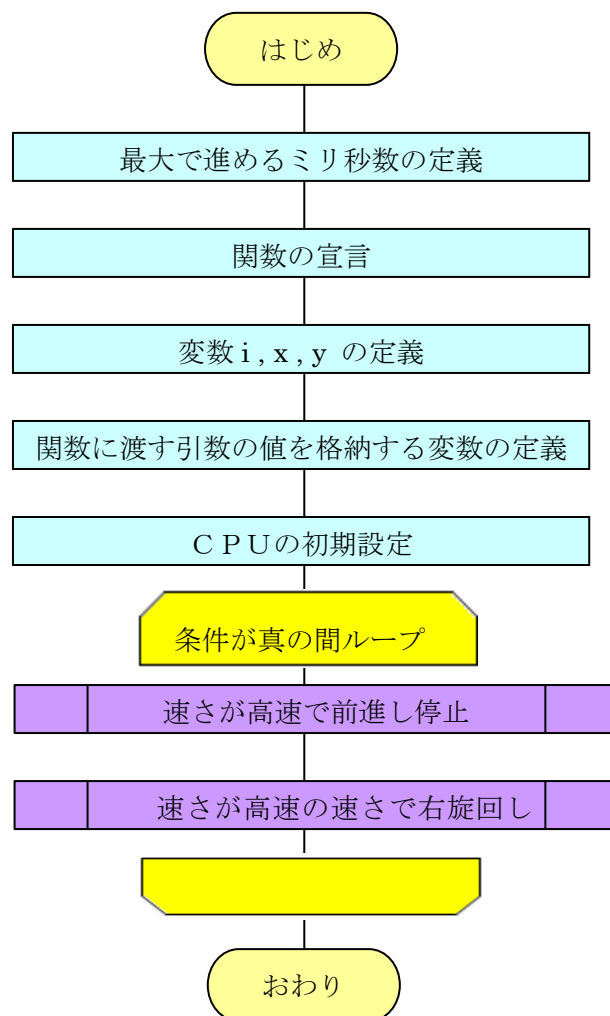



図 2-26

応用問題

配布した「サンプルプログラム　モータ制御」の一部を使用してスイッチを押したらロボットがダンスするようなプログラムを作成せよ．

[解答例]

```
[1]#define MAX_TIME 5000                                    //最大に進めるミリ秒
[2]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[3]void back_high(int sec,int x,double y,int i);
//速さが高速で後退し,走りすぎを制御する関数
[4]void R_turn_high(int sec,int x,double y,int i);
//速さが高速で右旋回し,走りすぎを制御する関数
[5]void L_turn_high(int sec,int x,double y,int i);
//速さが高速で左旋回し,走りすぎを制御する関数
[6]void R_turn2_high(int sec,int x,double y,int i);
//速さが高速でゆるやかに右旋回し,走りすぎを制御する関数
[7]void L_turn2_high(int sec,int x,double y,int i);
//速さが高速でゆるやかに左旋回し,走りすぎを制御する関数
[8]
[9]/*メイン関数*************************************************************/
[10]void main(void)
[11]{
[12]            int i;
[13]            int x;
[14]            double y;
[15]
[16]            int F_high=2000;         //速さが高速で前進するミリ秒を指定する
[17]            int B_high=3000;         //速さが高速で後退するミリ秒を指定する
[18]            int R_high=3500;         //速さが高速で右旋回するミリ秒を指定する
[19]            int L_high=3000;         //速さが高速で左旋回するミリ秒を指定する
[20]            int R2_high=2000;
//速さが高速でゆるやかに右旋回するミリ秒を指定する
[21]            int L2_high=1000;
//速さが高速でゆるやかに左旋回するミリ秒を指定する
[22]
[23]      //制御周期の設定[単位：Hz　範囲：30.0~]
[24]      const BYTE MainCycle = 60;
[25]      Init((BYTE)MainCycle);           //CPU の初期設定
[26]
[27]      //ループ
[28]      while(1){
[29]            if(getSW()){
[30]                  /*高速の速さで前進し停止する*/
[31]                  forward_high(F_high,x,y,i);               //関数の呼び出し
[32]
[33]                  /*高速の速さで左旋回し停止する*/
[34]                  L_turn_high(L_high,x,y,i);                //関数の呼び出し
[35]
[36]                  /*高速の速さでゆるやかに右旋回し停止する*/
```

```
[37]                    R_turn2_high(R2_high,x,y,i);              //関数の呼び出し
[38]
[39]                    /*高速の速さで後退し停止する*/
[40]                    back_high(B_high,x,y,i);                  //関数の呼び出し
[41]
[42]                    /*高速の速さで右旋回し停止する*/
[43]                    R_turn_high(R_high,x,y,i);                //関数の呼び出し
[44]
[45]                    /*高速の速さで前進し停止する*/
[46]                    forward_high(F_high,x,y,i);               //関数の呼び出し
[47]
[48]                    /*高速の速さでゆるやかに左旋回し停止する*/
[49]                    L_turn2_high(L_high,x,y,i);               //関数の呼び出し
[50]
[51]                    /*高速の速さで右旋回し停止する*/
[52]                    R_turn_high(R_high,x,y,i);                //関数の呼び出し
[53]
[54]                    /*高速の速さで後退し停止する*/
[55]                    back_high(B_high,x,y,i);                  //関数の呼び出し
[56]            }
[57]    }
[58]}
[59]
[60]/*****************************************************************************
[61]                    関数名：速さが高速で前進し,走りすぎを制御する関数
[62]                    引数:sec　進む時間(ミリ秒)
[63]                         x      割り算した結果の整数部分
[64]                         y      割り算した余り
[65]                         i      ループ用カウンタ変数
[66]***************************************************************************** /
[67]void forward_high(int sec,int x,double y,int i)
[68]{
[69]    if(sec<=MAX_TIME){
[70]    Mtr_Run(64,-64,0,0);                      //高速で前進
[71]    Wait(sec);                                //入力したミリ秒数だけ進む
[72]    Mtr_Run(0,0,0,0);                         //停止
[73]    LED(2);                                   //オレンジ色の LED 点灯
[74]    Wait(1000);                               //1 秒間待つ
[75]    }
[76]    else{
[77]    x=sec/MAX_TIME;
[78]    y=sec%MAX_TIME;
[79]    for(i=0;i<x;i++){
[80]            Mtr_Run(64,-64,0,0);              //高速で前進
[81]            Wait(MAX_TIME);                   //MAX_TIME ミリ秒待つ
[82]            Mtr_Run(0,0,0,0);                 //停止
[83]            LED(1);                           //緑色の LED 点灯
[84]            Wait(1000);                       //1 秒間待つ
[85]            }
```

```
[86]	Mtr_Run(64,-64,0,0);				//高速で前進
[87]	Wait(y);					//y ミリ秒待つ
[88]	Mtr_Run(0,0,0,0);				//停止
[89]	LED(3);						//全 LED 点灯
[90]	Wait(1000);					//1 秒間待つ
[91]	}
[92]}
[93]
[94]
[95]/*****************************************************************************
[96]			関数名：速さが高速で後退し,走りすぎを制御する関数
[97]			引数:sec　進む時間(ミリ秒)
[98]				x	割り算した結果の整数部分
[99]				y	割り算した余り
[100]				i	ループ用カウンタ変数
[101]***************************************************************************** /
[102]void back_high(int sec,int x,double y,int i)
[103]{
[104]	if(sec<=MAX_TIME){
[105]	Mtr_Run(-64,64,0,0);				//高速で後退
[106]	Wait(sec);					//入力したミリ秒数だけ進む
[107]	Mtr_Run(0,0,0,0);				//停止
[108]	LED(2);						//オレンジ色の LED 点灯
[109]	Wait(1000);					//1 秒間待つ
[110]	}
[111]	else{
[112]	x=sec/MAX_TIME;
[113]	y=sec%MAX_TIME;
[114]	for(i=0;i<x;i++){
[115]		Mtr_Run(-64,64,0,0);			//高速で後退
[116]		Wait(MAX_TIME);				//MAX_TIME ミリ秒待つ
[117]		Mtr_Run(0,0,0,0);			//停止
[118]		LED(1);					//緑色の LED 点灯
[119]		Wait(1000);				//1 秒間待つ
[120]		}
[121]	Mtr_Run(-64,64,0,0);				//高速で後退
[122]	Wait(y);					//y ミリ秒待つ
[123]	Mtr_Run(0,0,0,0);				//停止
[124]	LED(3);						//全 LED 点灯
[125]	Wait(1000);					//1 秒間待つ
[126]	}
[127]}
[128]
[129]/*****************************************************************************
[130]			関数名：速さが高速で右旋回し,走りすぎを制御する関数
[131]				引数:sec　進む時間(ミリ秒)
[132]					x	割り算した結果の整数部分
[133]					y	割り算した余り
[134]					i	ループ用カウンタ変数
```

```
[135]************************************************************************** /
[136]void R_turn_high(int sec,int x,double y,int i)
[137]{
[138]	if(sec<=MAX_TIME){
[139]	Mtr_Run(0,-64,0,0);				//高速で右旋回
[140]	Wait(sec);					//入力したミリ秒数だけ進む
[141]	Mtr_Run(0,0,0,0);				//停止
[142]	LED(2);						//オレンジ色の LED 点灯
[143]	Wait(1000);					//1 秒間待つ
[144]	}
[145]	else{
[146]	x=sec/MAX_TIME;
[147]	y=sec%MAX_TIME;
[148]	for(i=0;i<x;i++){
[149]		Mtr_Run(0,-64,0,0);			//高速で右旋回
[150]		Wait(MAX_TIME);				//MAX_TIME ミリ秒待つ
[151]		Mtr_Run(0,0,0,0);			//停止
[152]		LED(1);					//緑色の LED 点灯
[153]		Wait(1000);				//1 秒間待つ
[154]		}
[155]	Mtr_Run(0,-64,0,0);				//高速で右旋回
[156]	Wait(y);					//y ミリ秒待つ
[157]	Mtr_Run(0,0,0,0);				//停止
[158]	LED(3);						//全 LED 点灯
[159]	Wait(1000);					//1 秒間待つ
[160]	}
[161]}
[162]
[163]/***************************************************************************
[164]			関数名：速さが高速で左旋回し,走りすぎを制御する関数
[165]				引数:sec	進む時間(ミリ秒)
[166]					x	割り算した結果の整数部分
[167]					y	割り算した余り
[168]					i	ループ用カウンタ変数
[169]************************************************************************** /
[170]void L_turn_high(int sec,int x,double y,int i)
[171]{
[172]	if(sec<=MAX_TIME){
[173]	Mtr_Run(64,0,0,0);					//高速で左旋回
[174]	Wait(sec);						//入力したミリ秒数だけ進む
[175]	Mtr_Run(0,0,0,0);					//停止
[176]	LED(2);							//オレンジ色の LED 点灯
[177]	Wait(1000);						//1 秒間待つ
[178]	}
[179]	else{
[180]	x=sec/MAX_TIME;
[181]	y=sec%MAX_TIME;
[182]	for(i=0;i<x;i++){
[183]		Mtr_Run(64,0,0,0);				//高速で左旋回
```

```
[184]		Wait(MAX_TIME);				//MAX_TIME ミリ秒待つ
[185]		Mtr_Run(0,0,0,0);				//停止
[186]		LED(1);					//緑色の LED 点灯
[187]		Wait(1000);				//1 秒間待つ
[188]		}
[189]	Mtr_Run(64,0,0,0);				//高速で左旋回
[190]	Wait(y);					//y ミリ秒待つ
[191]	Mtr_Run(0,0,0,0);				//停止
[192]	LED(3);					//全 LED 点灯
[193]	Wait(1000);				//1 秒間待つ
[194]	}
[195]}
[196]
[197]/*****************************************************************************
[198]		関数名：速さが高速でゆるやかに右旋回し,走りすぎを制御する関数
[199]				引数:sec　進む時間(ミリ秒)
[200]				x	割り算した結果の整数部分
[201]				y	割り算した余り
[202]				 i	ループ用カウンタ変数
[203]***************************************************************************** /
[204]void R_turn2_high(int sec,int x,double y,int i)
[205]{
[206]	if(sec<=MAX_TIME){
[207]	Mtr_Run(32,-64,0,0);				//高速でゆるやかに右旋回
[208]	Wait(sec);					//入力したミリ秒数だけ進む
[209]	Mtr_Run(0,0,0,0);				//停止
[210]	LED(2);					//オレンジ色の LED 点灯
[211]	Wait(1000);				//1 秒間待つ
[212]	}
[213]	else{
[214]	x=sec/MAX_TIME;
[215]	y=sec%MAX_TIME;
[216]	for(i=0;i<x;i++){
[217]		Mtr_Run(32,-64,0,0);			//高速でゆるやかに右旋回
[218]		Wait(MAX_TIME);			//MAX_TIME ミリ秒待つ
[219]		Mtr_Run(0,0,0,0);			//停止
[220]		LED(1);				//緑色の LED 点灯
[221]		Wait(1000);			//1 秒間待つ
[222]		}
[223]	Mtr_Run(32,-64,0,0);				//高速でゆるやかに右旋回
[224]	Wait(y);					//y ミリ秒待つ
[225]	Mtr_Run(0,0,0,0);				//停止
[226]	LED(3);					//全 LED 点灯
[227]	Wait(1000);				//1 秒間待つ
[228]	}
[229]}
[230]
[231]/*****************************************************************************
[232]		関数名：速さが高速でゆるやかに左旋回し,走りすぎを制御する関数
```

```
[233]                                          引数:sec   進む時間(ミリ秒)
[234]                                                   x       割り算した結果の整数部分
[235]                                                   y       割り算した余り
[236]                                                   i       ループ用カウンタ変数
[237]***************************************************************************** /
[238]void L_turn2_high(int sec,int x,double y,int i)
[239]{
[240]     if(sec<=MAX_TIME){
[241]     Mtr_Run(64,-32,0,0);                    //高速でゆるやかに左旋回
[242]     Wait(sec);                              //入力したミリ秒数だけ進む
[243]     Mtr_Run(0,0,0,0);                       //停止
[244]     LED(2);                                 //オレンジ色の LED 点灯
[245]     Wait(1000);                             //1 秒間待つ
[246]     }
[247]     else{
[248]     x=sec/MAX_TIME;
[249]     y=sec%MAX_TIME;
[250]     for(i=0;i<x;i++){
[251]             Mtr_Run(64,-32,0,0);            //高速でゆるやかに左旋回
[252]             Wait(MAX_TIME);                 //MAX_TIME ミリ秒待つ
[253]             Mtr_Run(0,0,0,0);               //停止
[254]             LED(1);                         //緑色の LED 点灯
[255]             Wait(1000);                     //1 秒間待つ
[256]             }
[257]     Mtr_Run(64,-32,0,0);                    //高速でゆるやかに左旋回
[258]     Wait(y);                                //y ミリ秒待つ
[259]     Mtr_Run(0,0,0,0);                       //停止
[260]     LED(3);                                 //全 LED 点灯
[261]     Wait(1000);                             //1 秒間待つ
[262]     }
[263]}
```

[プログラム解説]

・1 行目は最大で進めるミリ秒数を定義しています.

・2〜7 行目は関数の宣言をしています．今回使用する関数は，速さが高速で前進し，走りすぎを制御する関数，速さが高速で後退し，走りすぎを制御する関数，速さが高速で右旋回し，走りすぎを制御する関数，速さが高速で左旋回し，走りすぎを制御する関数，速さが高速でゆるやかに右旋回し，走りすぎを制御する関数，速さが高速でゆるやかに左旋回し，走りすぎを制御する関数です.

・10 行目は関数の宣言です.

・12 行目はループ用カウンタの変数を宣言しています.

・13 行目は関数に実引数として変数 x を渡すための int 型の変数の宣言をしています.

・14 行目は関数に実引数として変数 y を渡すための double 型の変数の宣言をしています.

・16 行目は速さが高速で前進するミリ秒数を指定します．この値は forward_high 関数の引数の値として渡されます.

・17 行目は速さが高速で後退するミリ秒数を指定します．この値は back_high 関数の引数の値として渡されます.
・18 行目は速さが高速で右旋回するミリ秒数を指定します．この値は R_turn_high 関数の引数の値として渡されます.
・19 行目は速さが高速で左旋回するミリ秒数を指定します．この値は L_turn_high 関数の引数の値として渡されます.
・20 行目は速さが高速でゆるやかに右旋回するミリ秒数を指定します．この値は R_turn2_high 関数の引数の値として渡されます.
・21 行目は速さが高速でゆるやかに左旋回するミリ秒数を指定します．この値は L_turn2_high 関数の引数の値として渡されます.
・23～24 行目はＣＰＵの初期設定を行っています.
・28～57 行目のメインループ内では関数を呼び出しています.
・60～66 行目は関数名と引数の説明を行っています.
・67 行目は関数の宣言を行っています．関数の型は void 型です.
・69～75 行目はもし 16 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，16 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.
・76～91 行目まではもし 16 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 79 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.
・95～101 行目は関数名と引数の説明をしています.
・102 行目は関数の宣言を行っています．関数の型は void 型です.
・104～110 行目はもし 17 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，17 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.
・111～126 行目まではもし 17 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 114 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.
・129～135 行目は関数名と引数の説明をしています.
・136 行目は関数の宣言を行っています．関数の型は void 型です.
・138～144 行目はもし 18 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，18 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.
・145～160 行目まではもし 18 行目で指定したミリ秒数より MAX_TIME の値のほうが小

さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 148 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・163～169 行目は関数名と引数の説明をしています.

・170 行目は関数の宣言を行っています．関数の型は void 型です.

・172～178 行目はもし 19 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，19 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.

・179～194 行目まではもし 19 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 182 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・197～203 行目は関数名と引数の説明をしています.

・204 行目は関数の宣言を行っています．関数の型は void 型です.

・206～212 行目はもし 20 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，20 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.

・213～228 行目まではもし 20 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 216 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・231～237 行目は関数名と引数の説明をしています.

・238 行目は関数の宣言を行っています．関数の型は void 型です.

・240～246 行目はもし 21 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，21 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが 1 秒間停止したときに点灯するというものです.

・247～262 行目まではもし 21 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 250 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

☆フローチャート☆

図 2-27

高速の速さで前進し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-28

高速の速さで後退し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で後退
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で後退
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で後退
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり
図 2-29

高速の速さで右旋回し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-30

高速の速さで左旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で左旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で左旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で左旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-31

高速の速さゆるやかに右旋回し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速でゆるやかに右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速でゆるやかに右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速でゆるやかに右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-32

高速の速さゆるやかに左旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速でゆるやかに左旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速でゆるやかに左旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速でゆるやかに左旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-33

2.5. 赤外線センサを活用しよう

VS_WRC003 にはセンサを接続できるアナログ端子が四つ備わっており，さまざまなセンサを接続することができます．ＣＰＵボードには，接続したセンサにより出力電圧（アナログ値）を取得しそれをＡ／Ｄ変換して数値化します．この数値は 0～1024（12 ビット）の範囲で，0 が 0[V]1024 が 3.3[V]に対応します．

ここでは，学習用ロボットキット「BeautoChaser」に付属している赤外線センサの値を取得し，ＬＥＤを光らせてみましょう．

Ａ／Ｄ変換

A/D 変換とはアナログデータをデジタルデータに置き換える操作のことです．

ほとんどのマイコンでは，任意の電圧をアナログデータとして取得して処理できます．そのため，これに合わせて，市販のセンサも情報を電圧で出力するものが多く存在します．

電圧の大きさは 0～3.3[V]のように幅があるため，2 進数の数値のように「0」と「1」で表すことができません．そのため，マイコンでは 0～255 のように数値の桁を増やして電圧値を表現します．VS_WRC003 の場合，0～3.3[V]のアナログデータを 0～1024（12 ビット）デジタルデータに変換しています．1024 が 3.3[V]に対応していますので，1 が約 3.2[V]になります．

アナログ入力データを取得するには関数 AdRead（）を使用します．

○UNIT AdRead（BYTE ch）;

AdRead（）ではアナログ入力端子に入力された電圧を取得します．引数に ch に 0～3 を指定することで，必要な端子の電圧を取得できます．(0～3 の数値が，それぞれの基板上の AN1～4 の端子に相当します)．入力された電圧はデジタル数値に変換され UNIT 型（符号無し整数）で 0～1024 の値が戻り値として返されます．

[引数]

BYTE ch:取得するアナログ入力の端子番号を指定（0～3）

[戻り値]

アナログ入力端子にかかっている電圧を取得．範囲は 0～1024（0～3.3V）

センサの入力関数を使ったプログラムの main（）関数を以下に示します．

このプログラムでは赤外線センサの入力を取得し，その数値に応じてブザーの音程が変化します．BeautoChaser でこのプログラムを実行してみて，赤外線センサを手でふさいだりしたときにブザーの音程が変化するのか確認しましょう．

[解答例]

```
[1]void main(void)
[2]{
[3]	//制御周期の設定[単位：Hz　範囲：30.0~]
[4]	const BYTE MainCycle = 60;
[5]
[6]	Init((BYTE)MainCycle);		//CPU の初期設定
```

```
[7]
[8]         LED(3);                              //全 LED 点灯
[9]
[10]        //ループ
[11]        while(1){
[12]        int R_Senser=AdRead(1);              //右センサの値を取得し、変数に代入
[13]        Sync();                              //同期関数
[14]
[15]        if(R_Senser<216){
[16]                BuzzerSet(230,0x80);
[17]                BuzzerStart();               //ブザーが鳴り始める
[18]        }
[19]        else{
[20]                BuzzerSet(110,0x80);
[21]                BuzzerStart();               //ブザーが鳴り始める
[22]                }
[23]        }
[24]}
```

[プログラム解説]

・1 行目は関数の宣言です

・4～6 行目はＣＰＵの初期設定を行っています.

・8 行目はＬＥＤが全点灯します．この宣言をしなくてもＬＥＤが全点灯する仕組みになっています.

・10 行目以降はメインループです.

・11 行目から 23 行目は，取得したアナログ値が 216 より小さかったら音程が低い音が鳴るようにし，それ以外だったら音程が高い音が鳴るように設定しています

[フローチャート]

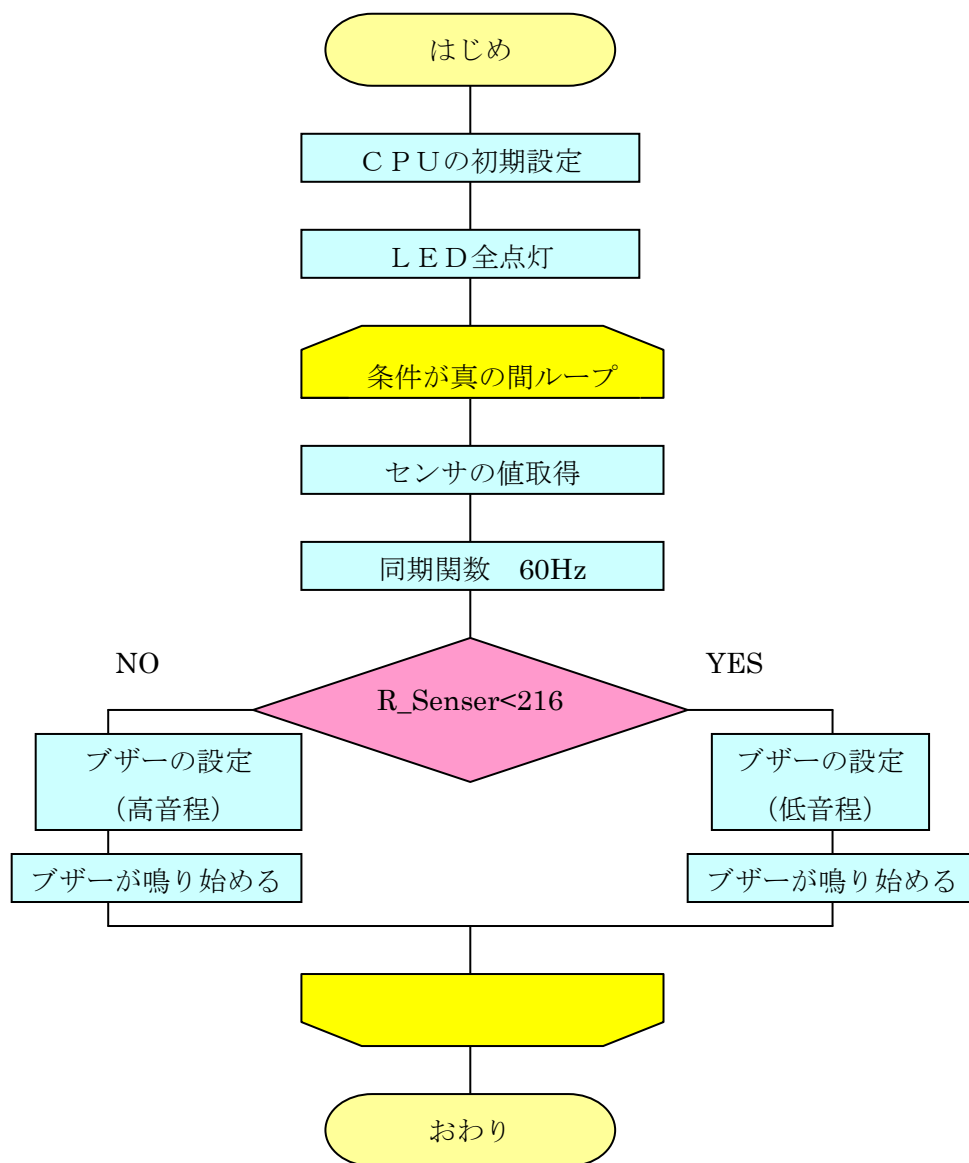



図 2-34

例題 1

センサの値によってＬＥＤの光り方とブザーの音程が変化するプログラムを作成しよう．

[解答例]

```
[1]void main(void)
[2]{
[3]		//制御周期の設定[単位：Hz　範囲：30.0~]
[4]		const BYTE MainCycle = 60;
[5]
[6]		Init((BYTE)MainCycle);				//CPU の初期設定
[7]		BuzzerSet(0x80,0x80);				//ブザーの初期設定
[8]
[9]		//ループ
[10]	while(1){
[11]	int data=AdRead(0);				//センサの値を取得し,data に代入
[12]	Sync();						//同期関数
[13]	if(data<128){
[14]			LED(0);				//LED 全消灯
[15]			BuzzerSet(220,0x80);		//音程が"ド"で大きい音
[16]			BuzzerStart();
[17]	}
[18]	else if(data<256){
[19]			LED(1);				//緑の LED 点灯
[20]			BuzzerSet(196,0x80);		//音程が"レ"で大きい音
[21]			BuzzerStart();
[22]	}
[23]	else if(data<512){
[24]			LED(2);				//オレンジ色の LED 点灯
[25]			BuzzerSet(175,0x80);		//音程が"ミ"で大きい音
[26]			BuzzerStart();
[27]	}
[28]	else{
[29]			LED(3);				//LED 全点灯
[30]			BuzzerSet(165, 0x80);		//音程が"ファ"で大きい音
[31]			BuzzerStart();
[32]	}
[33]	}
[34]}
```

[プログラム解説]

・1 行目は関数の宣言です．

・3～7 行目は各機能の初期設定を行っています．

・10 行目以降はメインループです．

・11 行目は変数 data に 0 番目の端子（ＡＮ１の端子）のアナログ値を取得し代入しています．

・12 行目は同期関数（60Hz で実行）です．

・13～32 行目では，取得したアナログ値が 128 より小さかったらＬＥＤが全消灯し，ブザーが大きい音で"ド"を鳴らす．256 より小さかったらＬＥＤが緑色に点灯し，ブザーが大

きい音で"レ"を鳴らす．512 より小さかったら，ＬＥＤがオレンジ色に点灯し，ブザーが大きい音で"ミ"を鳴らす．それ以外だったらＬＥＤが全点灯し，ブザーが大きい音で"ファ"を鳴らす．

☆フローチャート☆

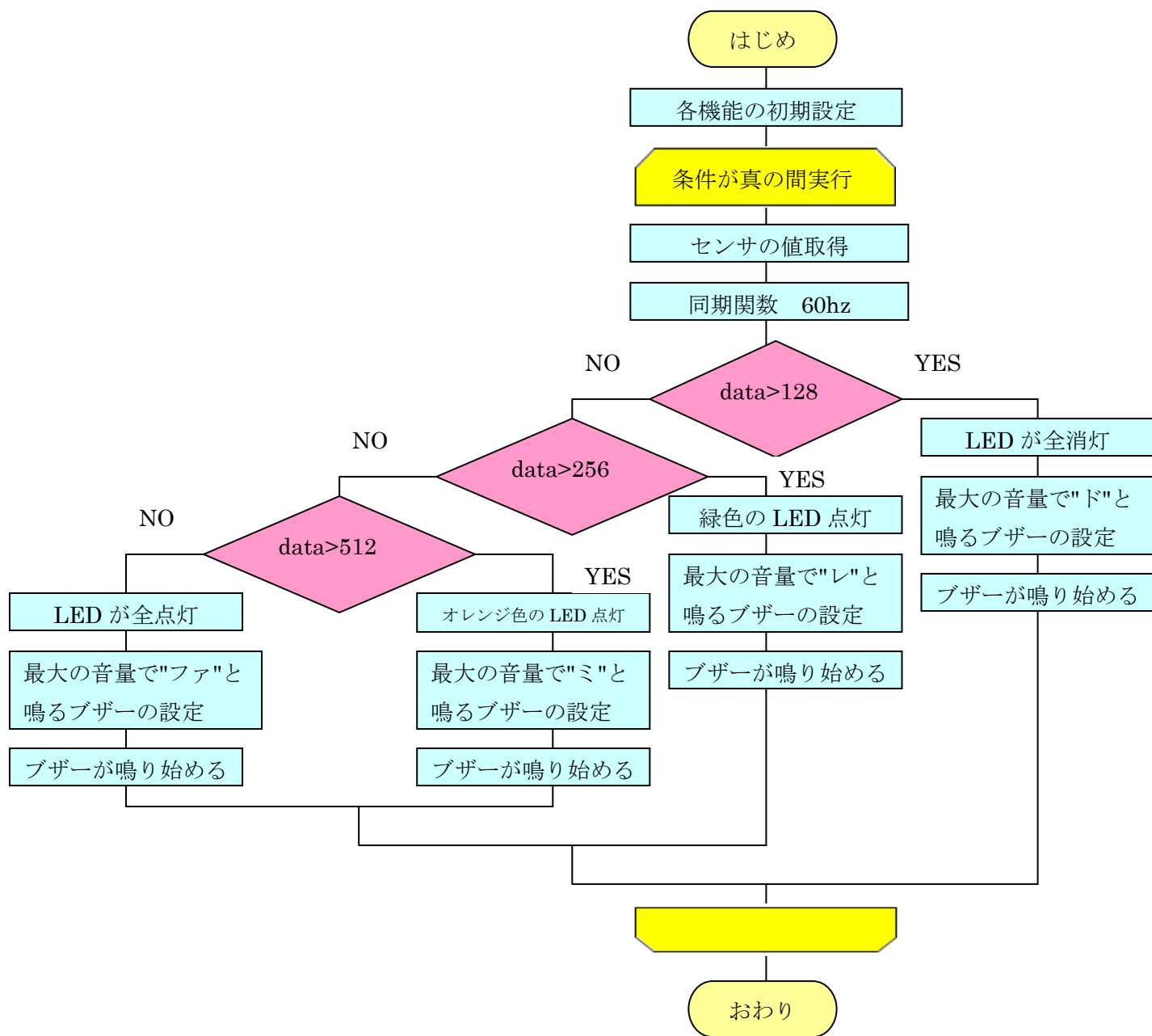



図 2-35

例題 2

センサの値でブザーの音程とボリュームが変化するプログラムを作成しよう.

[解答例]

```
[1]void main(void)
[2]{
[3]        //制御周期の設定[単位：Hz　範囲：30.0~]
[4]        const BYTE MainCycle = 60;
[5]        Init((BYTE)MainCycle);                    //CPU の初期設定
[6]
[7]        BuzzerSet(0,0);                           //ブザーの設定
[8]        LED(1);                                   //緑色の LED 点灯
[9]
[10]       while(1){
[11]               int L_Senser=AdRead(0);       //変数を宣言し,アナログ値に代入
[12]               int R_Senser=AdRead(1);       //変数を宣言し,アナログ値に代入
[13]               Sync();                       //同期関数　60Hz
[14]               BuzzerSet(L_Senser/5,R_Senser/5);      //ブザーの設定
[15]               BuzzerStart();
[16]       }
[17]}
```

[プログラム解説]

・1 行目は関数の宣言です.

・4～5 行目はＣＰＵの初期設定を行っています.

・7 行目はブザーの設定で音程と音量両方 0 にしています.

・8 行目はＬＥＤが緑色に点灯します.

・10 行目以降はメインループです.

・11 行目は左センサのアナログ値を変数に代入しています.

・12 行目は右センサのアナログ値を変数に代入しています.

・13 行目は同期関数（60Hz で実行）です.

・14 行目は注意が必要です．AdRead 関数で得られるアナログデータの数値と BuzzerSet の引数 L_Senser,R_Senser の数値とでは扱える数値の範囲が異なります．（前者は 0～1024，後者は 0～255）．AdRead 関数で得られた値をそのまま BuzzerSet 関数に与えると，数値の範囲をはみ出して意図しない動作になる可能性があります．これを防ぐために，アナログデータを取得した変数 data を除算して，数値の範囲を超えない値に収めています.

・15 行目はブザーが鳴り始めます.

☆フローチャート☆

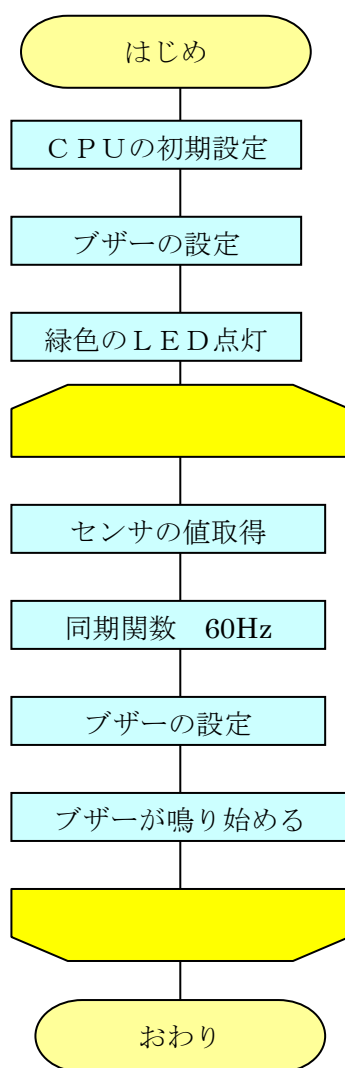



図 2-36

例題 3

配布した「サンプルプログラム　モータ制御」の一部を使用して，センサの値の変化によって前進やゆるやかに右旋回をして，ブザーが鳴るようなプログラムを作成せよ．

[解答例]

```
[1]#define MAX_TIME 5000                 //最大に進めるミリ秒数の指定
[2]void forward_middle(int sec,int x,double y,int i);
[3]//速さが中速で前進し,走りすぎを制御する関数
[4]
[5]void R_turn2_middle(int sec,int x,double y,int i);
[6]//速さが中速でゆるやかに右旋回し走りすぎを制御する関数
[7]/*メイン関数*****************************************************/
[8]void main(void)
[9]{
[10]        int i;                       //ループ用カウンタ
[11]        int x;                       //関数に実引数として変数 x を渡すための変数の宣言
[12]        double y;                    //関数に実引数として変数 y を渡すための変数の宣言
[13]
[14]        int F_middle=1500;                 //速さが中速で前進するミリ秒を指定する
[15]        int R2_middle=2000;//速さが中速でゆるやかに右旋回するミリ秒を指定する
[16]
[17]        //制御周期の設定[単位：Hz　範囲：30.0~]
[18]        const BYTE MainCycle = 60;
[19]        Init((BYTE)MainCycle);                          //CPU の初期設定
[20]        BuzzerSet(0x80,0x80);                           //ブザーの設定
[21]        LED(2);                                         //オレンジ色の LED 点灯
[22]
[23]        while(1){
[24]                int L_Senser=AdRead(0); //左のセンサの値を取得
[25]                Sync();                                         //同期関数
[26]                if(L_Senser<256){
[27]                        /*中速の速さで前進し停止する*/
[28]                        forward_middle(F_middle,x,y,i);         //関数の呼び出し
[29]                        BuzzerSet(110,0x80);                    //ブザーの設定
[30]                        BuzzerStart();                  //ブザーを鳴らし始める
[31]            }
[32]                else{
[33]                        /*中速の速さでゆるやかに右旋回し停止する*/
[34]                        R_turn2_middle(R2_middle,x,y,i);        //関数の呼び出し
[35]                        BuzzerSet(220,0x80);                    //ブザーの設定
[36]                        BuzzerStart();                  //ブザーを鳴らし始める
[37]        }
[38]        }
[39]}
[40]
[41]/***************************************************************************
[42]                関数名：速さが中速で前進し,走りすぎを制御する関数
[43]                        引数:sec　進む時間(ミリ秒)
[44]                                x       割り算した結果の整数部分
```

```
[45]                                    y       割り算した余り
[46]                                    i       ループ用カウンタ変数
[47]****************************************************************************/
[48]void forward_middle(int sec,int x,double y,int i)
[49]{
[50]      if(sec<=MAX_TIME){
[51]      Mtr_Run(32,-32,0,0);                    //中速で前進
[52]      Wait(sec);                              //入力したミリ秒数だけ進む
[53]      Mtr_Run(0,0,0,0);                       //停止
[54]      LED(2);                                 //オレンジ色の LED 点灯
[55]      Wait(1000);                             //1 秒間待つ
[56]      }
[57]      else{
[58]      x=sec/MAX_TIME;
[59]      y=sec%MAX_TIME;
[60]      for(i=0;i<x;i++){
[61]              Mtr_Run(32,-32,0,0);            //中速で前進
[62]              Wait(MAX_TIME);                 //MAX_TIME ミリ秒待つ
[63]              Mtr_Run(0,0,0,0);               //停止
[64]              LED(1);                         //緑色の LED 点灯
[65]              Wait(1000);                     //1 秒間待つ
[66]              }
[67]      Mtr_Run(32,-32,0,0);                    //中速で前進
[68]      Wait(y);                                //y ミリ秒待つ
[69]      Mtr_Run(0,0,0,0);                       //停止
[70]      LED(3);                                 //全 LED 点灯
[71]      Wait(1000);                             //1 秒間待つ
[72]      }
[73]}
[74]
[75]/****************************************************************************
[76]      関数名：速さが中速でゆるやかに右旋回し,走りすぎを制御する関数
[77]                      引数:sec      進む時間(ミリ秒)
[78]                              x     割り算した結果の整数部分
[79]                              y     割り算した余り
[80]                               i      ループ用カウンタ変数
[81]****************************************************************************/
[82]void R_turn2_middle(int sec,int x,double y,int i)
[83]{
[84]      if(sec<=MAX_TIME){
[85]      Mtr_Run(16,-32,0,0);                    //中速でゆるやかに右旋回
[86]      Wait(sec);                              //入力したミリ秒数だけ進む
[87]      Mtr_Run(0,0,0,0);                       //停止
[88]      LED(2);                                 //オレンジ色の LED 点灯
[89]      Wait(1000);                             //1 秒間待つ
[90]      }
[91]      else{
[92]      x=sec/MAX_TIME;
[93]      y=sec%MAX_TIME;
```

```
[94]    for(i=0;i<x;i++){
[95]            Mtr_Run(16,-32,0,0);                //中速でゆるやかに右旋回
[96]            Wait(MAX_TIME);                     //MAX_TIME ミリ秒待つ
[97]            Mtr_Run(0,0,0,0);                   //停止
[98]            LED(1);                             //緑色の LED 点灯
[99]            Wait(1000);                         //1 秒間待つ
[100]           }
[101]   Mtr_Run(16,-32,0,0);                        //中速でゆるやかに右旋回
[102]   Wait(y);                                    //y ミリ秒待つ
[103]   Mtr_Run(0,0,0,0);                           //停止
[104]   LED(3);                                     //全 LED 点灯
[105]   Wait(1000);                                 //1 秒間待つ
[106]   }
[107]}
```

[プログラム解説]

・1 行目は最大で進めるミリ秒数を指定します.

・2～6 行目はそれぞれの関数の宣言をしています.

・8 行目は関数の宣言をしています.

・10 行目はループカウンタ用の変数を宣言しています.

・11 行目は関数に実引数として変数 x を渡すための変数の宣言をしています

・12 行目は関数に実引数として変数 y を渡すための変数の宣言をしています.

・14 行目は forward_middle という関数に実引数として渡す，速さが中速で前進するミリ秒を指定する変数を宣言します.

・15 行目は R_turn2_middle という関数に実引数として渡す，速さが中速でゆるやかに右旋回するミリ秒を指定する変数を宣言しています.

・17～19 行目はＣＰＵの初期設定を行っています.

・20 行目はブザーの初期設定を行っています.

・21 行目はオレンジ色のＬＥＤが点灯しています.

・23 行目以降はメインループです.

・24 行目では変数 L_Senser に 0 番目の端子（ＡＮ１の端子）のアナログ値を代入しています.

・25 行目は同期関数（60Hz で実行）です.

・26～37 行目までは，取得したアナログ値が 256 より小さかったら，中速の速さで前進しながらブザーを鳴らします．それ以外なら中速でゆるやかに右旋回しながらブザーを鳴らします．ここではそれぞれ関数を呼び出しています.

・41～47 行目は関数名と引数の説明をしています.

・48～73 行目は関数の宣言をしています．関数内は以下で説明します.

・50～56 行目はもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，14 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.

・57～72 行目まではもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 60 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.
・75～81 行目は関数名と引数の説明をしています.
・82～107 行目は関数の宣言をしています．関数内は以下で説明します.
・84～90 行目はもし 15 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，15 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.
・57～72 行目まではもし 15 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 94 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

☆フローチャート☆

図 2-37

中速の速さで前進し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
中速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
中速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
中速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-38

中速の速さでゆるやかに右旋回し停止
NO
YES
sec<=MAX_TIME
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
中速でゆるやかに右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
中速でゆるやかに右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
中速でゆるやかに右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 2-39

3. C 言語プログラミングの応用編にチャレンジ！！

3.1. 赤外線センサで光走行性ロボットを体験しよう

光走性とは，生物が外部からの光（刺激）に反応し，光源に向かって移動する現象のことでBeautoChaserのCPUボードには出荷時に赤外線センサを使ってこの現象を模したプログラムが書き込まれています．ここでは，今まで学習した要素を使って C 言語で光走性ロボットのプログラミングをします．

図 3-2，図 3-3，図 3-4 に光走性プログラミングのフローチャートを示します．まず，光がどの方向にあるのか探さなければなりません. BeautoChaserには赤外線センサが 2 つありますが図 3-1 ように前に取り付けることによって光走性ロボットを体験することが出来ます．光を探索中に光を見つけたら（センサの値がある値＝しきい値より小さくなったら）前進し，光源に近づきます．

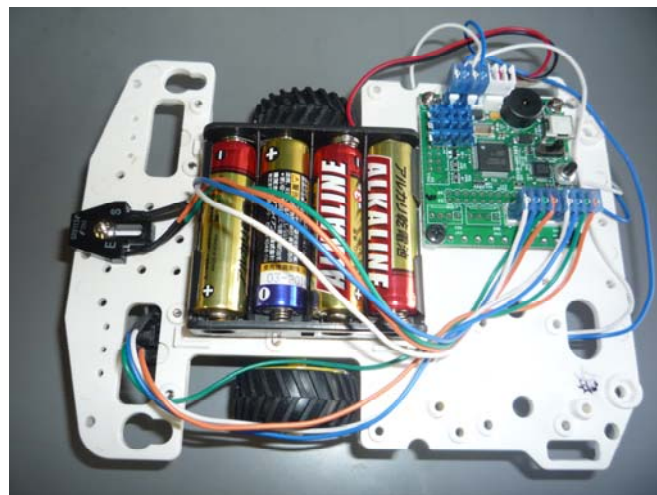



図 3-1　BeautoChaser

図 3-2，図 3-3，図 3-4 のフローチャートをプログラムにすると，以下のようになります．

[解答例]

```
[1]#define MAX_TIME 5000                       //最大に進めるミリ秒
[2]
[3]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[4]void turn_high(int sec,int x,double y,int i);
//速さが高速で旋回し，走りすぎを制御する
[5]
[6]void main(void)
[7]{
[8]        int i;                    //ループ用カウンタ
[9]        int x;                    //関数に実引数として変数 x を渡すための変数の宣言
[10]       double y;                 //関数に実引数として変数 y を渡すための変数の宣言
[11]       int data;
[12]
```

```
[13]	int F_high=4000;		//速さが高速で前進するミリ秒を指定する
[14]	int T_high=4000;		//速さが高速で旋回するミリ秒を指定する
[15]
[16]	//制御周期の設定[単位：Hz　範囲：30.0~]
[17]	const BYTE MainCycle = 60;
[18]	Init((BYTE)MainCycle);			//CPU の初期設定
[19]	LED(2);					//オレンジ色の LED 点灯
[20]
[21]	BuzzerSet(0x80,0x80);			//ブザーの設定
[22]
[23]	//ループ
[24]	while(1){
[25]		Sync();
[26]		data=AdRead(0);			//センサの値を取得し、data に代入
[27]
[28]		if(getSW()){			//スイッチが押されたら
[29]		Mtr_Run(0,0,0,0);		//モータの停止
[30]		}
[31]		else if(data<512)		//センサの値が 512 以下だったら
[32]		{
[33]		/*高速の速さで前進し停止する*/
[34]		forward_high(F_high,x,y,i);		//関数の呼び出し
[35]		BuzzerSet(0x80,0x80);			//ブザーの設定
[36]		BuzzerStart();				//ブザーが鳴り始める
[37]		Wait(1000);				//1 秒間待つ
[38]		BuzzerStop();				//ブザーが止まる
[39]		}
[40]		else{					//それ以外
[41]		/*高速の速さで旋回し停止する*/
[42]		turn_high(T_high,x,y,i);		//関数の呼び出し
[43]		BuzzerSet(0x80,0x80);			//ブザーの設定
[44]		BuzzerStart();				//ブザーが鳴り始める
[45]		Wait(1000);				//1 秒間待つ
[46]		BuzzerStop();				//ブザーが止まる
[47]		}
[48]	}
[49]}
[50]
[51]/**************************************************************************
[52]		関数名：速さが高速で前進し,走りすぎを制御する関数
[53]			引数:sec　進む時間(ミリ秒)
[54]				x	割り算した結果の整数部分
[55]				y	割り算した余り
[56]				i	ループ用カウンタ変数
[57]************************************************************************** /
[58]void forward_high(int sec,int x,double y,int i)
[59]{
[60]	if(sec<=MAX_TIME){
[61]	Mtr_Run(64,-64,0,0);			//高速で前進
```

```
[62]	Wait(sec);	//入力したミリ秒数だけ進む
[63]	Mtr_Run(0,0,0,0);	//停止
[64]	LED(2);	//オレンジ色の LED 点灯
[65]	Wait(1000);	//1 秒間待つ
[66]	}
[67]	else{
[68]	x=sec/MAX_TIME;
[69]	y=sec%MAX_TIME;
[70]	for(i=0;i<x;i++){
[71]		Mtr_Run(64,-64,0,0);	//高速で前進
[72]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[73]		Mtr_Run(0,0,0,0);	//停止
[74]		LED(1);	//緑色の LED 点灯
[75]		Wait(1000);	//1 秒間待つ
[76]		}
[77]	Mtr_Run(64,-64,0,0);	//高速で前進
[78]	Wait(y);	//y ミリ秒待つ
[79]	Mtr_Run(0,0,0,0);	//停止
[80]	LED(3);	//全 LED 点灯
[81]	Wait(1000);	//1 秒間待つ
[82]	}
[83]}
[84]
[85]/******************************************************************************
[86]		関数名：速さが高速で旋回し,走りすぎを制御する関数
[87]			引数:sec　進む時間(ミリ秒)
[88]				x	割り算した結果の整数部分
[89]				y	割り算した余り
[90]				i	ループ用カウンタ変数
[91]****************************************************************************** /
[92]void turn_high(int sec,int x,double y,int i)
[93]{
[94]	if(sec<=MAX_TIME){
[95]	Mtr_Run(64,64,0,0);	//高速で旋回
[96]	Wait(sec);	//入力したミリ秒数だけ進む
[97]	Mtr_Run(0,0,0,0);	//停止
[98]	LED(2);	//オレンジ色の LED 点灯
[99]	Wait(1000);	//1 秒間待つ
[100]	}
[101]	else{
[102]	x=sec/MAX_TIME;
[103]	y=sec%MAX_TIME;
[104]	for(i=0;i<x;i++){
[105]		Mtr_Run(64,64,0,0);	//高速で旋回
[106]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[107]		Mtr_Run(0,0,0,0);	//停止
[108]		LED(1);	//緑色の LED 点灯
[109]		Wait(1000);	//1 秒間待つ
[110]		}
```

```
[111]      Mtr_Run(64,64,0,0);                    //高速で旋回
[112]      Wait(y);                               //y ミリ秒待つ
[113]      Mtr_Run(0,0,0,0);                      //停止
[114]      LED(3);                                //全 LED 点灯
[115]      Wait(1000);                            //1 秒間待つ
[116]      }
[117]}
```

［プログラム解説］

・1 行目は最大に進めるミリ秒を定義します

・3～4 行目はそれぞれの関数のプロトタイプ宣言をします．

・8～11 行目はそれぞれの変数の宣言をします．

・13 行目は速さが高速で前進するミリ秒を指定する，

・14 行目は速さが高速で旋回するミリ秒を指定する．

・17～18 行目は CPU の初期設定を行っています．

・19 行目はオレンジ色の LED が点灯します．

・21 行目はブザーの初期設定を行っています．

・24 行目以降はメインループ内にはいります．

・25 行目は同期関数（60Hz で実行）です．

・26 行目は左センサの値を取得し，data に代入します．

・28～49 行目はもしスイッチが押されたら停止，センサの値が 1000 より小さかったら前進する関数を呼び出しブザーが 1 秒間鳴り続け，それ以外だったら旋回する関数を呼び出し，ブザーが 1 秒間鳴り続けます．

・51～57 行目は関数名と引数の説明をしています．

・58～83 行目は関数の宣言をしています．関数内は以下で説明します．

・60～66 行目はもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，13 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです．

・67～82 行目まではもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 70 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します．

・85～91 行目は関数名と引数の説明をしています．

・93～117 行目は関数の宣言をしています．関数内は以下で説明します．

・94～100 行目はもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，14 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです．

・101～116 行目まではもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME

を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 104 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します．

POINT

上記のプログラム 31 行目で，赤外線センサのしきい値を設定しています．

If(data<512){

ここでは 512 に設定していますが，ロボットを動かす場所の明るさやセンサの取り付け方によって，この条件式を満たさずロボットがラインに反応しないことがあります．

うまく反応しない場合は，しきい値を変えてみたり，センサの取りつけ方（地面からの距離・角度）を変えながら試してみましょう．

Key　Word

しきい値

センサを扱うプログラムで，ある数値を境にして状態を区別する際の基準となる数値を「しきい値」といいます．上記の光走性の問題では，センサが「光を見つけた」と見なすしきい値として「512」を設定しています．

☆フローチャート☆

はじめ
最大ミリ秒数の定義
関数の宣言
変数 i , x , y,data の定義
関数に渡す引数の値を格納する変数の定義
ＣＰＵの初期設定
オレンジ色のＬＥＤ点灯
同期関数 60Hz
左センサの値取得
NO
YES
スイッチが
押されたか？
停止
YES
NO
光を
見つけたか？
速さが高速で
前進し停止
速さが高速で
旋回し停止
ブザーの設定
ブザーの設定
ブザーが鳴り始める
ブザーが鳴り始める
１秒間待つ
１秒間待つ
ブザーが止まる
ブザーが止まる
おわり

図 3-2

高速の速さで前進し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-3

高速の速さで旋回し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-4

POINT

赤外線センサ

本書にたびたび登場する赤外線センサの正式名称は，反射型フォトセンサといいます．反射型フォトセンサは以下の図のように発光部（LED）と受光部（フォトダイオード）を持ち，発光部からの光を物体が反射し，その光を受光部で受け取ることで，正面にある物体の有無を判断することができます．

また，認識した物体の色が白いか黒いかによって，反射して戻ってくる LED の光の強さが変わるため，物体における色の濃淡も判断できます．この方法を利用したのが後で出てくるライントレースです．

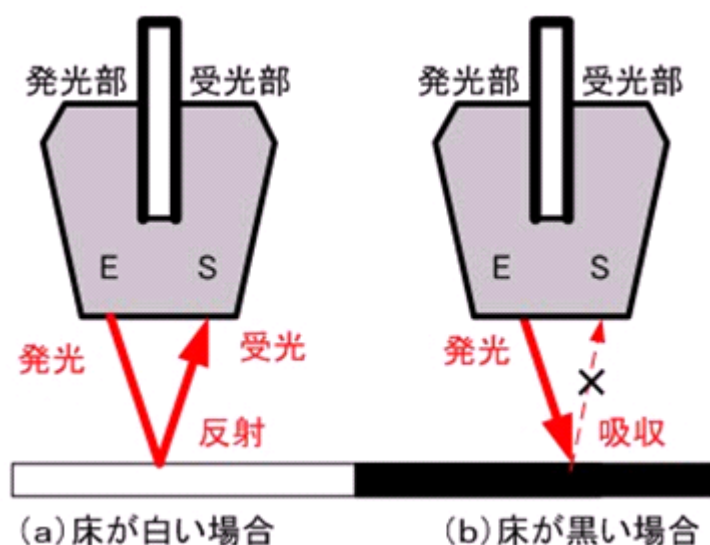



図 3-5　赤外線センサ

光走性ロボットの場合，反射してくる LED の光ではなく，ロボットに照らされる光を直接センサの受光部で認識することで，光源がどこにあるのか探すことが出来ます．

3.2. ライントレースロボットを体験しよう

ライントレースとは床面に引かれた線（床が白い場合は黒，黒い場合は白いライン）にそって移動するロボットです．ライントレースを競技に盛り込んだ大会は全国各地で開催されており，世界的な競技会のロボカップジュニアレスキューや，マイコンカーラリーが有名です．

BeautoChaser でライントレースを行う場合，以下の手順に沿って赤外線センサを床に向けて取りつけます．

手順①　ブラケットへの取りつけ

ブラケットに赤外線センサをネジ止めしてください．（ネジは**図 3-6** のものを使います．）表記されている E（エミッタ），S（センサ）を図 3-8 の位置にくるように取りつけてください．

図 3-6　ネジ

[引用：H8 マイコンによる組込みプログラミング入門 p.78]

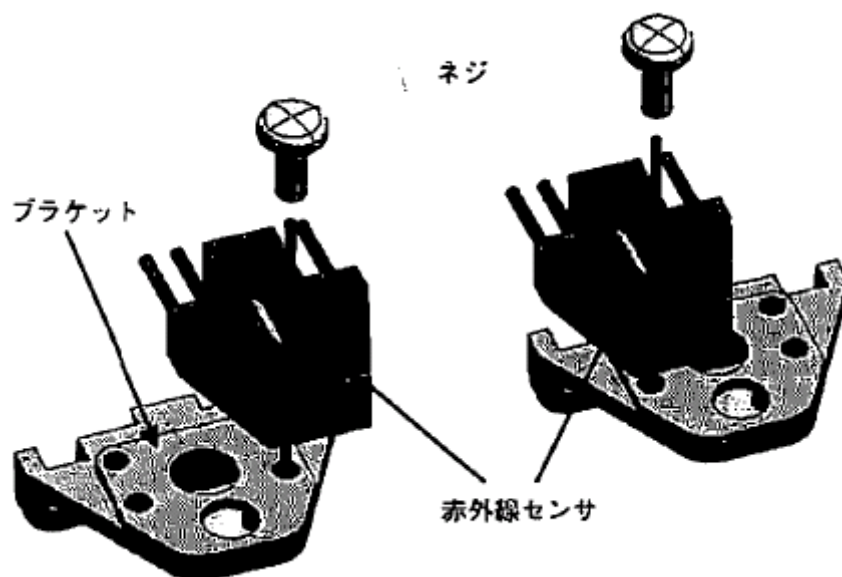



図 3-7　赤外線センサの取りつけ①

[引用：H8 マイコンによる組込みプログラミング入門 p.78]

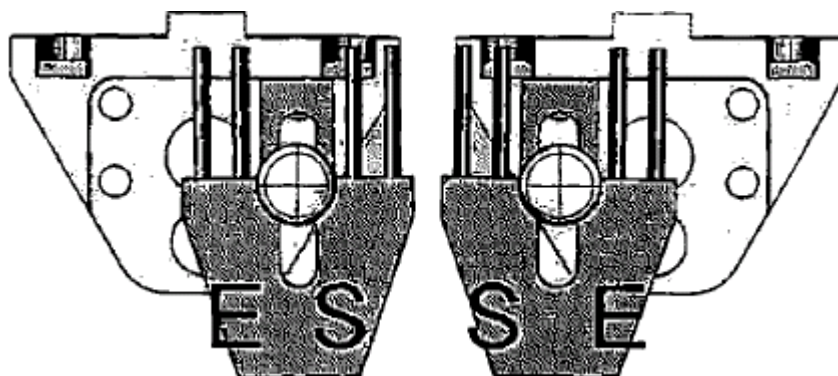



図 3-8　赤外線センサの向き

[引用：H8 マイコンによる組込みプログラミング入門 p.78]

手順②　ベースへの取りつけ

ベースに左右の赤外線センサを各 2 本のネジで止めしてください．左右の赤外線センサが内側に位置するように取りつけてください．

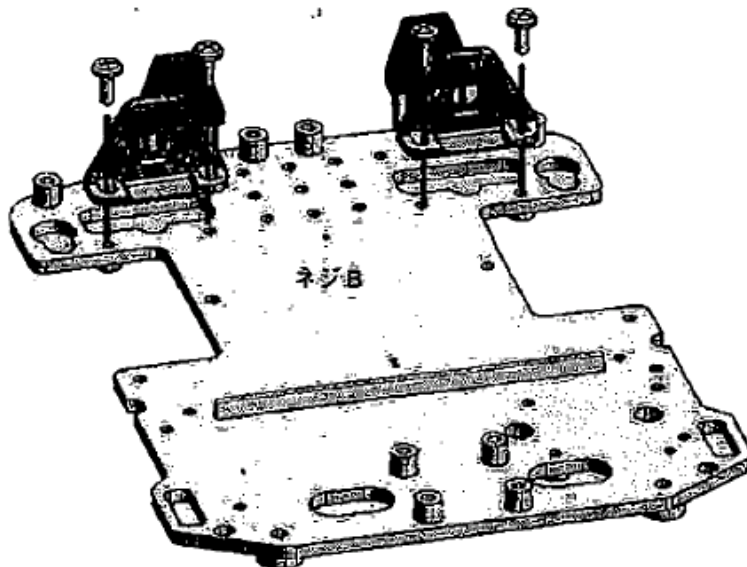



図 3-9　赤外線センサの取りつけ②

[引用：H8 マイコンによる組込みプログラミング入門 p.79]

手順③　センサを内側に少し傾けると面を認識しやすくなります．

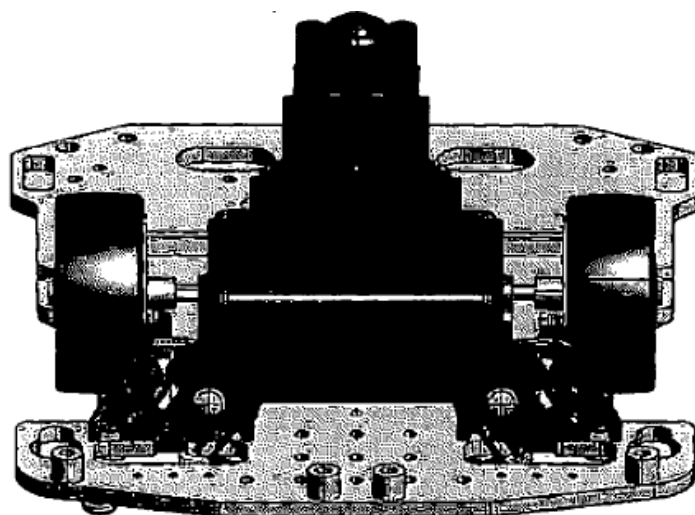



図 3-10　センサの追加が完了

[引用：H8 マイコンによる組込みプログラミング入門 p.79]

手順④　左センサを VS-WRC003 上の AN1 のコネクタに，右のセンサを AN2 のコネクタにそれぞれ接続します．

それでは，ライントレースする方法を考えてみましょう．

まず，二つのセンサの間にラインを挟んだ状態で本体をライン上におきます．この状態ではラインは確実にセンサの間にあるので前進してもラインを外れません．前進しているロボットが図 3-11 のような左カーブに差し掛かったとき，左のセンサは黒いラインを検出します．このときは左のタイヤを停止させ左に旋回します．右向きのカーブのときも同様に，カーブのある右のタイヤを停止させて右旋回します．しばらく旋回すると，再びラインが二つのセンサの間に入り，センサはラインを検出していない状態になりますので，再び前進します．

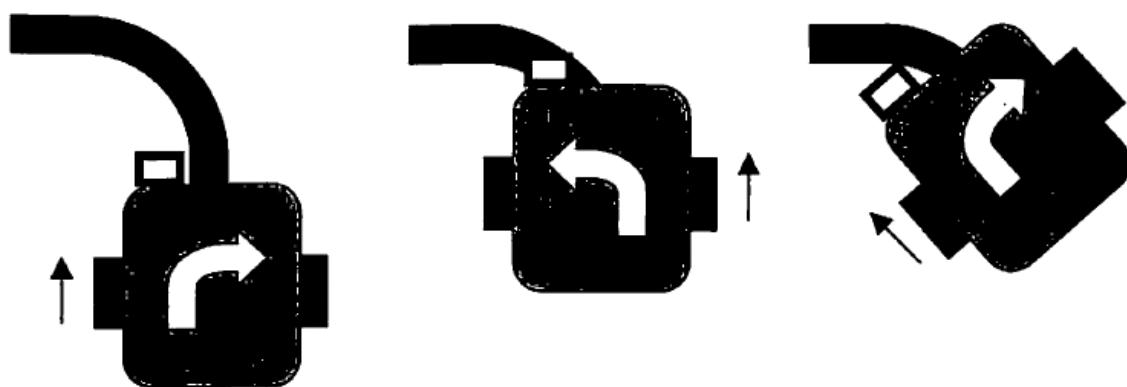



図 3-11　ラインの検出の流れ

[引用：H8 マイコンによる組込みプログラミング入門 p.80]

POINT

センサ

センサは自然現象や人工物の機械的・電磁気的・熱的・音響的・化学的性質あるいは，それらで示される空間情報・時間情報を，何らかの化学的原理を応用して，人間や機械が扱いやすい別媒体の信号に置き換える装置である．

POINT

ライントレース

ライントレースとは本文にもあるとおり，床面に引かれたラインをロボットがセンサを利用して読み取り，ラインに沿って走行することです．例えば，工場で部品を自動的に運ぶロボットなどにこの技術が利用されています．

ライントレースする方法としては大きく分けて 2 種類，ラインのエッジ（はし）を認識し，そのエッジに沿って走行する方法と，ラインそのものが床面のどの位置にあるかを認識する方法があります．エッジを認識する場合は，ラインのエッジを中心にロボットのセンサ部分を左右に振りながら走行し，ライン上ならライン外，逆にセンサがライン外ならライン上へ戻るような動きでライン上をなぞります．ラインそのものを認識する場合には複数のセンサを利用してラインの位置を検出し，車体の中心を基準点とし，ラインが左右どちらかにずれていないかを判断しながら走行します．ライントレースしている様子を図 3-12 に示します．

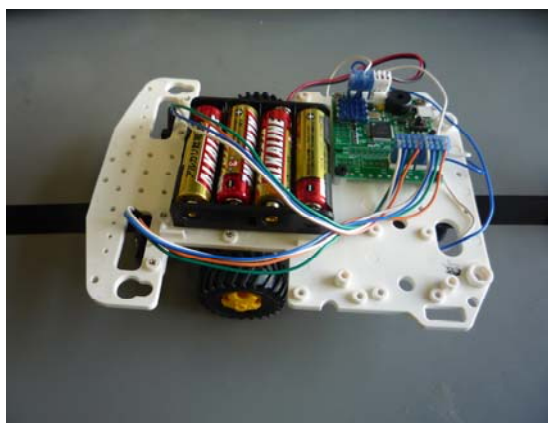



図 3-12　ライントレース

次にコースを準備しましょう．白い床に黒いビニールテープコースで作ってみましょう．注意する点としてカーブの形は図 3-13 の左のような鋭角のカーブがもっとも曲がるのが難しく，右の緩やかなカーブのほうが簡単に曲がることができます．はじめは緩やかなカーブでコースを書いたほうがクリアしやすくなります．

図 3-13　カーブの難易度（右に行くほど曲がるのは簡単）

それでは，ライントレースする方法を考えてみましょう．

まず，センサが反応していないときは，右タイヤのみを前進させ左に前進しながら曲がります．そのまま曲がり続けると，センサが黒いラインを検出します．黒いラインを検出したときは，左タイヤのみを前進させながら曲がります．これを繰り返すことで，ジグザグにラインにそって進むことができます．

[注意事項]

ライントレースなど赤外線センサを使った問題に取り組む際には，光が入り込まないように紙等で赤外線センサの周りを囲み，光がなるべく入り込まないようにしましょう．

例題 1

配布した「サンプルプログラム　モータ制御」の一部を使用して，図 3-15 のような渦巻状のラインにそってライントレースするプログラムを作成せよ．

（渦巻きのラインのカーブはなるべく緩やかにしましょう）

図 3-14

[解答例]

```
[1]#define MAX_TIME 5000                                    //最大に進めるミリ秒
[2]void L_turn_high(int sec,int x,double y,int i);
//速さが高速で左旋回し,走りすぎを制御する関数
[3]void R_turn_high(int sec,int x,double y,int i);
//速さが高速で右旋回し,走りすぎを制御する関数
[4]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[5]
[6]void main(void)
```

```
[7]{
[8]	int i;						//ループ用カウンタ
[9]	int x;				//関数に実引数として変数 x を渡すための変数の宣言
[10]	double y;			//関数に実引数として変数 y を渡すための変数の宣言
[11]
[12]	int L_high=1000;		//速さが高速で左旋回するミリ秒を指定する
[13]	int R_high=2000;		//速さが高速で右旋回するミリ秒を指定する
[14]	int F_high=1500;		//速さが高速で前進するミリ秒を指定する
[15]
[16]	//制御周期の設定[単位：Hz　範囲：30.0~]
[17]	const BYTE MainCycle = 60;
[18]
[19]	Init((BYTE)MainCycle);		//CPU の初期設定
[20]	Mtr_Run(0,0,0,0);		//停止
[21]	LED(3);				//オレンジの LED 点灯
[22]
[23]	while(1){
[24]		int L_Senser=AdRead(0);			//左センサの値取得
[25]		int R_Senser=AdRead(1);			//右センサの値取得
[26]		Sync();
[27]		if(L_Senser>512){
[28]			/*高速の速さで左旋回し停止する*/
[29]			L_turn_high(L_high,x,y,i);		//関数の呼び出し
[30]			LED(1);					//緑色の LED 点灯
[31]		}
[32]	else if(R_Senser>512){
[33]			/*高速の速さで右旋回し停止する*/
[34]			R_turn_high(R_high,x,y,i);		//関数の呼び出し
[35]			LED(2);				//オレンジ色の LED 点灯
[36]		}else{
[37]			/*高速の速さで前進し停止する*/
[38]			forward_high(F_high,x,y,i);		//関数の呼び出し
[39]			LED(3);						//全点灯
[40]		}
[41]	}
[42]}
[43]
[44]/*****************************************************************************
[45]		関数名：速さが高速で左旋回し,走りすぎを制御する関数
[46]				引数:sec　進む時間(ミリ秒)
[47]					x	割り算した結果の整数部分
[48]					y	割り算した余り
[49]					i	ループ用カウンタ変数
[50]***************************************************************************** /
[51]void L_turn_high(int sec,int x,double y,int i)
[52]{
[53]	if(sec<=MAX_TIME){
[54]	Mtr_Run(64,0,0,0);			//高速で左旋回
[55]	Wait(sec);				//入力したミリ秒数だけ進む
```

```
[56]	Mtr_Run(0,0,0,0);	//停止
[57]	LED(2);	//オレンジ色の LED 点灯
[58]	Wait(1000);	//1 秒間待つ
[59]	}
[60]	else{
[61]	x=sec/MAX_TIME;
[62]	y=sec%MAX_TIME;
[63]	for(i=0;i<x;i++){
[64]		Mtr_Run(64,0,0,0);	//高速で左旋回
[65]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[66]		Mtr_Run(0,0,0,0);	//停止
[67]		LED(1);	//緑色の LED 点灯
[68]		Wait(1000);	//1 秒間待つ
[69]		}
[70]	Mtr_Run(64,0,0,0);	//高速で左旋回
[71]	Wait(y);	//y ミリ秒待つ
[72]	Mtr_Run(0,0,0,0);	//停止
[73]	LED(3);	//全 LED 点灯
[74]	Wait(1000);	//1 秒間待つ
[75]	}
[76]}
[77]/****************************************************************************
[78]		関数名：速さが高速で右旋回し,走りすぎを制御する関数
[79]			引数:sec　進む時間(ミリ秒)
[80]				x	割り算した結果の整数部分
[81]				y	割り算した余り
[82]				i	ループ用カウンタ変数
[83]**************************************************************************** /
[84]void R_turn_high(int sec,int x,double y,int i)
[85]{
[86]	if(sec<=MAX_TIME){
[87]	Mtr_Run(0,-64,0,0);	//高速で右旋回
[88]	Wait(sec);	//入力したミリ秒数だけ進む
[89]	Mtr_Run(0,0,0,0);	//停止
[90]	LED(2);	//オレンジ色の LED 点灯
[91]	Wait(1000);	//1 秒間待つ
[92]	}
[93]	else{
[94]	x=sec/MAX_TIME;
[95]	y=sec%MAX_TIME;
[96]	for(i=0;i<x;i++){
[97]		Mtr_Run(0,-64,0,0);	//高速で右旋回
[98]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[99]		Mtr_Run(0,0,0,0);	//停止
[100]		LED(1);	//緑色の LED 点灯
[101]		Wait(1000);	//1 秒間待つ
[102]		}
[103]	Mtr_Run(0,-64,0,0);	//高速で右旋回
[104]	Wait(y);	//y ミリ秒待つ
```

```
[105]     Mtr_Run(0,0,0,0);                                   //停止
[106]     LED(3);                                             //全 LED 点灯
[107]     Wait(1000);                                         //1 秒間待つ
[108]     }
[109]}
[110]/**************************************************************************
[111]                    関数名：速さが高速で前進し,走りすぎを制御する関数
[112]                           引数:sec   進む時間(ミリ秒)
[113]                                x      割り算した結果の整数部分
[114]                                y      割り算した余り
[115]                                 i      ループ用カウンタ変数
[116]**************************************************************************/
[117]void forward_high(int sec,int x,double y,int i)
[118]{
[119]     if(sec<=MAX_TIME){
[120]     Mtr_Run(64,-64,0,0);                        //高速で前進
[121]     Wait(sec);                                  //入力したミリ秒数だけ進む
[122]     Mtr_Run(0,0,0,0);                           //停止
[123]     LED(2);                                     //オレンジ色の LED 点灯
[124]     Wait(1000);                                 //1 秒間待つ
[125]     }
[126]     else{
[127]     x=sec/MAX_TIME;
[128]     y=sec%MAX_TIME;
[129]     for(i=0;i<x;i++){
[130]             Mtr_Run(64,-64,0,0);                //高速で前進
[131]             Wait(MAX_TIME);                     //MAX_TIME ミリ秒待つ
[132]             Mtr_Run(0,0,0,0);                   //停止
[133]             LED(1);                             //緑色の LED 点灯
[134]             Wait(1000);                         //1 秒間待つ
[135]             }
[136]     Mtr_Run(64,-64,0,0);                        //高速で前進
[137]     Wait(y);                                    //y ミリ秒待つ
[138]     Mtr_Run(0,0,0,0);                           //停止
[139]     LED(3);                                     //全 LED 点灯
[140]     Wait(1000);                                 //1 秒間待つ
[141]     }
[142]}
```

[プログラム解説]

・1 行目は最大に進めるミリ秒を定義します.

・2～4 行目はそれぞれの関数のプロトタイプ宣言をします.

・8～10 行目はそれぞれの変数の宣言をしています.

・12 行目は速さが高速で左旋回するミリ秒数を指定しています.

・13 行目は速さが高速で右旋回するミリ秒数を指定しています.

・14 行目は速さが高速で前進するミリ秒を指定する.

・16～19 行目は CPU の初期設定を行っています.

・20 行目は停止しています.

・21 行目は LED がオレンジ色に点灯します.

・23 行目以降はメインループ内です.

・24～25 行目は左センサと右センサの値をそれぞれ取得しています.

・26 行目は同期関数（60Hz で実行）です.

・27～40 行目はもし左センサの値が 512 より大きかったら, 低速の速さで左旋回し停止し, 緑色の LED が点灯します. もし, 右センサの値が 512 より大きかったら右旋回し停止し, オレンジ色の LED を点灯します. それ以外だったら中速で前進し, LED が全点灯します.

・44～50 行目は関数名と引数の説明をしています.

・51～76 行目は関数の宣言をしています. 関数内は以下で説明します.

・53～59 行目はもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら, モータが 12 行目で指定したミリ秒数だけ進み, オレンジ色のＬＥＤに点灯し, 1 秒間停止するというものです.

・60～75 行目まではもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより, 割り算した結果だけ 63 行目の for 文内の処理を繰り返し, for 文が終了したら y ミリ秒作動します.

・77～83 行目は関数名と引数の説明をしています.

・84～109 行目は関数の宣言をしています. 関数内は以下で説明します.

・86～92 行目はもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら, 13 行目で指定したミリ秒数だけ進み, オレンジ色のＬＥＤが点灯し, 1 秒間停止するというものです.

・93～108 行目まではもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより, 割り算した結果だけ 96 行目の for 文内の処理を繰り返し, for 文が終了したら y ミリ秒作動します.

・110～116 行目は関数名と引数の説明をしています.

・117～142 行目は関数の宣言をしています. 関数内は以下で説明します.

・119～125 行目はもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら, 14 行目で指定したミリ秒数だけ進み, オレンジ色のＬＥＤが点灯し, 1 秒間停止するというものです.

・126～141 行目まではもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより, 割り算した結果だけ 129 行目の for 文内の処理を繰り返し, for 文が終了したら y ミリ秒作動します.

☆フローチャート☆

図 3-15

高速の速さで左旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で左旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で左旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で左旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-16

高速の速さで右旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-17

高速の速さで前進し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-18

例題 2

配布した「サンプルプログラム　モータ制御」の一部を使用して，図 3-19 のように白い大きな紙に黒いテープで囲んだものを使用し，ロボットが黒い線に近づいたら回避するプログラムを作成せよ．また回避する際にはブザーが鳴るようにプログラムを作成せよ．

図 3-19　　　　図 3-20

[解答例]

```
[1]#define MAX_TIME 5000                                   //最大に進めるミリ秒
[2]void L_turn_high(int sec,int x,double y,int i);
//速さが高速で左旋回し,走りすぎを制御する関数
[3]void R_turn_high(int sec,int x,double y,int i);
//速さが高速で右旋回し,走りすぎを制御する関数
[4]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[5]
[6]void main(void)
[7]{
[8]        int i;                                    //ループ用カウンタ
[9]        int x;                        //関数に実引数として変数 x を渡すための変数の宣言
[10]       double y;                     //関数に実引数として変数 y を渡すための変数の宣言
[11]
[12]       int L_high=500;                    //速さが高速で左旋回するミリ秒を指定する
[13]       int R_high=500;                    //速さが高速で右旋回するミリ秒を指定する
[14]       int F_high=500;                    //速さが高速で前進するミリ秒を指定する
[15]
[16]       //制御周期の設定[単位：Hz　範囲：30.0~]
[17]       const BYTE MainCycle = 60;
```

```
[18]
[19]     Init((BYTE)MainCycle);                    //CPU の初期設定
[20]     Mtr_Run(0,0,0,0);                         //停止
[21]     LED(3);                                   //オレンジの LED 点灯
[22]
[23]     while(1){
[24]             int L_Senser=AdRead(0);                   //左センサの値取得
[25]             int R_Senser=AdRead(1);                   //右センサの値取得
[26]             Sync();
[27]             if(L_Senser>512){
[28]                     /*高速の速さで左旋回し停止する*/
[29]                     L_turn_high(L_high,x,y,i);                //関数の呼び出し
[30]                     LED(1);                                   //緑色の LED 点灯
[31]                     BuzzerSet(220,0x80);                      //ブザー設定（ド）
[32]                     BuzzerStart();                        //ブザーが鳴り始める
[33]             }
[34]     else if(R_Senser>512){
[35]                     /*高速の速さで右旋回し停止する*/
[36]                     R_turn_high(R_high,x,y,i);                //関数の呼び出し
[37]                     LED(2);                           //オレンジ色の LED 点灯
[38]                     BuzzerSet(196,0x80);                      //ブザー設定（レ）
[39]                     BuzzerStart();                    //ブザーが鳴り始める
[40]             }else{
[41]                     /*高速の速さで前進し停止する*/
[42]                     forward_high(F_high,x,y,i);               //関数の呼び出し
[43]                     LED(3);                                   //全点灯
[44]             }
[45]     }
[46]}
[47]
[48]/**************************************************************************
[49]                     関数名：速さが高速で左旋回し,走りすぎを制御する関数
[50]                             引数:sec   進む時間(ミリ秒)
[51]                                   x     割り算した結果の整数部分
[52]                                   y     割り算した余り
[53]                                   i     ループ用カウンタ変数
```

```
[54]*****************************************************************************/
[55]void L_turn_high(int sec,int x,double y,int i)
[56]{
[57]	if(sec<=MAX_TIME){
[58]	Mtr_Run(64,0,0,0);				//高速で左旋回
[59]	Wait(sec);				//入力したミリ秒数だけ進む
[60]	Mtr_Run(0,0,0,0);				//モータの停止
[61]	LED(2);					//オレンジ色の LED 点灯
[62]	Wait(1000);				//1 秒間待つ
[63]	}
[64]	else{
[65]	x=sec/MAX_TIME;
[66]	y=sec%MAX_TIME;
[67]	for(i=0;i<x;i++){
[68]		Mtr_Run(64,0,0,0);			//高速で左旋回
[69]		Wait(MAX_TIME);			//MAX_TIME ミリ秒待つ
[70]		Mtr_Run(0,0,0,0);			//停止
[71]		LED(1);				//緑色の LED 点灯
[72]		Wait(1000);			//1 秒間待つ
[73]		}
[74]	Mtr_Run(64,0,0,0);				//高速で左旋回
[75]	Wait(y);					//y ミリ秒待つ
[76]	Mtr_Run(0,0,0,0);				//モータの停止
[77]	LED(3);					//全 LED 点灯
[78]	Wait(1000);				//1 秒間待つ
[79]	}
[80]}
[81]/*****************************************************************************
[82]			関数名：速さが高速で右旋回し,走りすぎを制御する関数
[83]				引数:sec　進む時間(ミリ秒)
[84]				    x	割り算した結果の整数部分
[85]				    y	割り算した余り
[86]				    i	ループ用カウンタ変数
[87]*****************************************************************************/
[88]void R_turn_high(int sec,int x,double y,int i)
[89]{
```

```
[90]    if(sec<=MAX_TIME){
[91]    Mtr_Run(0,-64,0,0);                 //高速で右旋回
[92]    Wait(sec);                          //入力したミリ秒数だけ進む
[93]    Mtr_Run(0,0,0,0);                   //停止
[94]    LED(2);                             //オレンジ色の LED 点灯
[95]    Wait(1000);                         //1 秒間待つ
[96]    }
[97]    else{
[98]    x=sec/MAX_TIME;
[99]    y=sec%MAX_TIME;
[100]   for(i=0;i<x;i++){
[101]           Mtr_Run(0,-64,0,0);         //高速で右旋回
[102]           Wait(MAX_TIME);             //MAX_TIME ミリ秒待つ
[103]           Mtr_Run(0,0,0,0);           //停止
[104]           LED(1);                     //緑色の LED 点灯
[105]           Wait(1000);                 //1 秒間待つ
[106]           }
[107]   Mtr_Run(0,-64,0,0);                 //高速で右旋回
[108]   Wait(y);                            //y ミリ秒待つ
[109]   Mtr_Run(0,0,0,0);                   //停止
[110]   LED(3);                             //全 LED 点灯
[111]   Wait(1000);                         //1 秒間待つ
[112]   }
[113]}
[114]/*****************************************************************************
[115]                   関数名：速さが高速で前進し,走りすぎを制御する関数
[116]                   引数:sec   進む時間(ミリ秒)
[117]                         x       割り算した結果の整数部分
[118]                         y       割り算した余り
[119]                         i       ループ用カウンタ変数
[120]***************************************************************************** /
[121]void forward_high(int sec,int x,double y,int i)
[122]{
[123]   if(sec<=MAX_TIME){
[124]   Mtr_Run(64,-64,0,0);                //高速で前進
[125]   Wait(sec);                          //入力したミリ秒数だけ進む
```

```
[126]	Mtr_Run(0,0,0,0);	//停止
[127]	LED(2);	//オレンジ色の LED 点灯
[128]	Wait(1000);	//1 秒間待つ
[129]	}
[130]	else{
[131]	x=sec/MAX_TIME;
[132]	y=sec%MAX_TIME;
[133]	for(i=0;i<x;i++){
[134]		Mtr_Run(64,-64,0,0);	//高速で前進
[135]		Wait(MAX_TIME);	//MAX_TIME ミリ秒待つ
[136]		Mtr_Run(0,0,0,0);	//停止
[137]		LED(1);	//緑色の LED 点灯
[138]		Wait(1000);	//1 秒間待つ
[139]		}
[140]	Mtr_Run(64,-64,0,0);	//高速で前進
[141]	Wait(y);	//y ミリ秒待つ
[142]	Mtr_Run(0,0,0,0);	//停止
[143]	LED(3);	//全 LED 点灯
[144]	Wait(1000);	//1 秒間待つ
[145]	}
[146]}
```

[プログラム解説]

・1 行目は最大に進めるミリ秒を定義します.
・2～4 行目はそれぞれの関数のプロトタイプ宣言をします.
・8～10 行目はそれぞれの変数の宣言をしています.
・12 行目は速さが高速で左旋回するミリ秒数を指定しています.
・13 行目は速さが高速で右旋回するミリ秒数を指定しています.
・14 行目は速さが高速で前進するミリ秒を指定する.
・16～19 行目は CPU の初期設定を行っています.
・20 行目はモータを停止しています.
・21 行目は LED がオレンジ色に点灯します.
・23 行目以降はメインループ内です.
・24～25 行目は左センサと右センサの値をそれぞれ取得しています.
・26 行目は同期関数（60Hz で実行）です.
・27～44 行目はもし左センサの値が 512 より大きかったら低速の速さで左旋回し停止し，緑色の LED が点灯し，ブザーの"ド"が鳴り始めます．もし，右センサの値が 512 より大

きかったら右旋回し停止し, オレンジ色の LED を点灯し, ブザーの"レ"が鳴り始めます. それ以外だったら中速で前進し，LED が全点灯します.

・48～54 行目は関数名と引数の説明をしています.

・55～80 行目は関数の宣言をしています. 関数内は以下で説明します.

・57～63 行目はもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，12 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.

・64～79 行目まではもし 12 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより，割り算した結果だけ 67 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・81～87 行目は関数名と引数の説明をしています.

・88～113 行目は関数の宣言をしています. 関数内は以下で説明します.

・90～96 行目はもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，13 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.

・97～112 行目まではもし 13 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより，割り算した結果だけ 100 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

・114～120 行目は関数名と引数の説明をしています.

・121～145 行目は関数の宣言をしています. 関数内は以下で説明します.

・123～129 行目はもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，14 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.

・130～145 行目まではもし 14 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら, 変数 x に sec と MAX_TIME を割り算した結果, 変数 y に sec と MAX_TIME を剰余した結果を代入します. これを求めることにより，割り算した結果だけ 133 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

☆フローチャート☆

図 3-21

高速の速さで左旋回し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で左旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で左旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で左旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-22

高速の速さで右旋回し停止
NO
sec<=MAX_TIME
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で右旋回
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で右旋回
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で右旋回
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-23

高速の速さで前進し停止
sec<=MAX_TIME
NO
YES
Sec/MAX_TIME の結果を x に代入
Sec%MAX_TIME の結果を y に代入
i<x になったら終了
高速で前進
MAX_TIME ミリ秒待つ
停止
緑色のＬＥＤ点灯
1 秒待つ
高速で前進
y ミリ秒待つ
停止
全ＬＥＤ点灯
1 秒待つ
高速で前進
sec ミリ秒待つ
停止
オレンジ色のＬＥＤ点灯
1 秒待つ
おわり

図 3-24

3.3. ランダム関数を使用しよう

C 言語では乱数を発生させることは出来ませんが，擬似乱数ならば使用する事ができます．乱数とは出現する数に規則性のない数です．例えばサイコロの目などがそれにあたります．擬似乱数とは乱数ではなく，計算によって乱数に近い数字を発生させる方法で，これを使ってランダムに動作するプログラムを作ってみましょう．乱数を発生させる関数はヘッダファイル stdlib.h で定義されていますので，これをインクルードします．乱数を発生させるにはまず関数 srand（）で発生系列を変更します．その後，関数 rand（）を実行することで，戻り値として擬似乱数が得られます．

○void srand（unsingde seed）

Srand（）は rand（）で発生させる擬似乱数の発生系列を変更する関数です．引数 seed に値を指定することで，発生系列を選択できます．Srand（）関数を実行すると，rand 関数を実行するたびに，その戻り値が乱数のようにさまざまな値に切り替わります．ただし，srand（）関数を引数 seed に毎回同じ値を指定すると，rand（）関数を実行するたびに得られる戻り値が毎回同じ順番で数値が変化するため，若干変化に乏しくなります．このため，srand 関数には，プログラムを実行するたびに異なる値が得られる仕組みが望ましいです．PC でのプログラミングには，現在の時間を返す time や「センサの数値」などを利用します．

[引数]

Unsined seed:擬似乱数の発生系列を変更する値

[戻り値]

なし

○int rand（void）

Rand（）関数には擬似乱数を発生させる関数で，0～RAND_MAX の範囲の擬似乱数を戻り値として返します．

[引数]

無し

[戻り値]

０～RAND_MAX の範囲の擬似乱数

応用問題

それでは，配布した「サンプルプログラム　モータ制御」の一部を使用して，図 3-25 のように囲まれた黒いラインから回避するプログラムを作成するにあたり，乱数を使用して

回避する角度がランダムに変更されるプログラムを作成しましょう.

図 3-25　　　　　　　　図 3-26

[解答例]

```
[1]#define MAX_TIME 5000                                    //最大に進めるミリ秒
[2]void forward_high(int sec,int x,double y,int i);
//速さが高速で前進し,走りすぎを制御する関数
[3]void main(void)
[4]{
[5]      int i;                                             //ループ用カウンタ
[6]      int x;                        //関数に実引数として変数 x を渡すための変数の宣言
[7]      double y;                     //関数に実引数として変数 y を渡すための変数の宣言
[8]
[9]      int F_high=500;                     //速さが高速で前進するミリ秒を指定する
[10]
[11]     //制御周期の設定[単位：Hz　範囲：30.0~]
[12]     const BYTE MainCycle = 60;
[13]     Init((BYTE)MainCycle);              //CPU の初期設定
[14]
[15]     Mtr_Run(0,0,0,0);                             //モータの停止
[16]     LED(3);                                       //全ての LED 点灯
[17]
[18]     while(1){
[19]             int m=0;                              //カウント用変数の宣言
[20]             int j=0;                              //カウント用変数の宣言
[21]             int result;                           //結果を代入するための変数
[22]             int L_Senser=AdRead(0);               //左センサの値取得
[23]             int R_Senser=AdRead(0);               //左センサの値取得
[24]
[25]             i=rand()%7;                           //0～7 までの乱数を発生させる
[26]             j=rand()%7;                           //0～7 までの乱数を発生させる
[27]
[28]             Sync();
[29]             if(L_Senser>512){
[30]                     m=m+1;
[31]                     srand(i);                     //変数 i の値を seed にする
```

```
[32]                result=rand()%7;          //0～7 までの乱数を発生させる
[33]                switch(result){           //result で条件分岐
[34]                case 0:                               //0 のとき
[35]                Mtr_Run(-64,64,0,0);                  //高速で後退
[36]                Wait(500);                            //0.5 秒待つ
[37]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[38]                LED(1);                               //緑色の LED 点灯
[39]                Wait(1000);                           //1 秒待つ
[40]                break;
[41]
[42]                case 1:                               //1 のとき
[43]                Mtr_Run(-64,64,0,0);                  //高速で後退
[44]                Wait(500);                            //0.5 秒待つ
[45]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[46]                LED(1);                               //緑色の LED 点灯
[47]                Wait(2000);                           //2 秒待つ
[48]                break;
[49]
[50]                case 2:                               //2 のとき
[51]                Mtr_Run(-64,64,0,0);                  //高速で後退
[52]                Wait(500);                            //0.5 秒待つ
[53]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[54]                LED(1);                               //緑色の LED 点灯
[55]                Wait(3000);                           //3 秒待つ
[56]                break;
[57]
[58]                case 3:                               //3 のとき
[59]                Mtr_Run(-64,64,0,0);                  //高速で後退
[60]                Wait(500);                            //0.5 秒待つ
[61]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[62]                LED(1);                               //緑色の LED 点灯
[63]                Wait(4000);                           //3 秒待つ
[64]                break;
[65]
[66]                case 4:                               //4 のとき
[67]                Mtr_Run(-64,64,0,0);                  //高速で後退
[68]                Wait(500);                            //0.5 秒待つ
[69]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[70]                LED(1);                               //緑色の LED 点灯
[71]                Wait(5000);                           //5 秒待つ
[72]                break;
[73]
[74]                case 5:                               //5 のとき
[75]                Mtr_Run(-64,64,0,0);                  //高速で後退
[76]                Wait(500);                            //0.5 秒待つ
[77]                Mtr_Run(64,0,0,0);                //右タイヤを前に(左旋回)
[78]                LED(1);                               //緑色の LED 点灯
[79]                Wait(6000);                           //6 秒待つ
[80]                break;
```

```
[81]
[82]                    case 6:                         //6 のとき
[83]                    Mtr_Run(-64,64,0,0);            //高速で後退
[84]                    Wait(500);                      //0.5 秒待つ
[85]                    Mtr_Run(64,0,0,0);          //右タイヤを前に(左旋回)
[86]                    LED(1);                         //緑色の LED 点灯
[87]                    Wait(7000);                     //7 秒待つ
[88]                    break;
[89]
[90]                    default:                        //それ以外のとき
[91]                    break;                      //なにもせずにブレーク
[92]                    }
[93]            }
[94]
[95]      else if(R_Senser>512){
[96]                    j=j+1;
[97]                    srand(j);                   //変数 j の値を seed にする
[98]                    result=rand()%7;        //0～7 までの乱数を発生させる
[99]
[100]                   switch(result){                 //result で条件分岐
[101]                   case 0:                         //0 のとき
[102]                   Mtr_Run(-64,64,0,0);            //高速で後退
[103]                   Wait(500);                      //0.5 秒待つ
[104]                   Mtr_Run(0,64,0,0);          //右タイヤを前に(右旋回)
[105]                   LED(1);                         //緑色の LED 点灯
[106]                   Wait(1000);                     //1 秒待つ
[107]                   break;
[108]
[109]                   case 1:                         //1 のとき
[110]                   Mtr_Run(-64,64,0,0);            //高速で後退
[111]                   Wait(500);                      //0.5 秒待つ
[112]                   Mtr_Run(0,64,0,0);          //右タイヤを前に(右旋回)
[113]                   LED(1);                         //緑色の LED 点灯
[114]                   Wait(2000);                     //2 秒待つ
[115]                   break;
[116]
[117]                   case 2:                         //2 のとき
[118]                   Mtr_Run(-64,64,0,0);            //高速で後退
[119]                   Wait(500);                      //0.5 秒待つ
[120]                   Mtr_Run(0,64,0,0);          //右タイヤを前に(右旋回)
[121]                   LED(1);                         //緑色の LED 点灯
[122]                   Wait(3000);                     //3 秒待つ
[123]                   break;
[124]
[125]                   case 3:                         //3 のとき
[126]                   Mtr_Run(-64,64,0,0);            //高速で後退
[127]                   Wait(500);                      //0.5 秒待つ
[128]                   Mtr_Run(0,64,0,0);          //右タイヤを前に(右旋回)
[129]                   LED(1);                         //緑色の LED 点灯
```

```
[130]                   Wait(4000);                         //3 秒待つ
[131]                   break;
[132]
[133]                   case 4:                             //4 のとき
[134]                   Mtr_Run(-64,64,0,0);                //高速で後退
[135]                   Wait(500);                          //0.5 秒待つ
[136]                   Mtr_Run(0,64,0,0);            //右タイヤを前に(右旋回)
[137]                   LED(1);                             //緑色の LED 点灯
[138]                   Wait(5000);                         //5 秒待つ
[139]                   break;
[140]
[141]                   case 5:                             //5 のとき
[142]                   Mtr_Run(-64,64,0,0);                //高速で後退
[143]                   Wait(500);                          //0.5 秒待つ
[144]                   Mtr_Run(0,64,0,0);            //右タイヤを前に(右旋回)
[145]                   LED(1);                             //緑色の LED 点灯
[146]                   Wait(6000);                         //6 秒待つ
[147]                   break;
[148]
[149]                   case 6:                             //6 のとき
[150]                   Mtr_Run(-64,64,0,0);                //高速で後退
[151]                   Wait(500);                          //0.5 秒待つ
[152]                   Mtr_Run(0,64,0,0);            //右タイヤを前に(右旋回)
[153]                   LED(1);                             //緑色の LED 点灯
[154]                   Wait(7000);                         //7 秒待つ
[155]                   break;
[156]
[157]                   default:                            //それ以外のとき
[158]                   break;
[159]                               }
[160]           }
[161]     else{
[162]                   /*高速の速さで前進し停止する*/
[163]                   forward_high(F_high,x,y,i);         //関数の呼び出し
[164]                   LED(3);                             //全点灯
[165]           }
[166]     }
[167]}
[168]
[169]/****************************************************************************
[170]                   関数名：速さが高速で前進し,走りすぎを制御する関数
[171]                         引数:sec   進む時間(ミリ秒)
[172]                              x      割り算した結果の整数部分
[173]                              y      割り算した余り
[174]                              i      ループ用カウンタ変数
[175]**************************************************************************** /
[176]void forward_high(int sec,int x,double y,int i)
[177]{
[178]     if(sec<=MAX_TIME){
```

```
[179]	Mtr_Run(64,-64,0,0);			//高速で前進
[180]	Wait(sec);				//入力したミリ秒数だけ進む
[181]	Mtr_Run(0,0,0,0);			//停止
[182]	LED(2);					//オレンジ色の LED 点灯
[183]	Wait(1000);				//1 秒間待つ
[184]	}
[185]	else{
[186]	x=sec/MAX_TIME;
[187]	y=sec%MAX_TIME;
[188]	for(i=0;i<x;i++){
[189]		Mtr_Run(64,-64,0,0);		//高速で前進
[190]		Wait(MAX_TIME);			//MAX_TIME ミリ秒待つ
[191]		Mtr_Run(0,0,0,0);		//停止
[192]		LED(1);				//緑色の LED 点灯
[193]		Wait(1000);			//1 秒間待つ
[194]		}
[195]	Mtr_Run(64,-64,0,0);			//高速で前進
[196]	Wait(y);				//y ミリ秒待つ
[197]	Mtr_Run(0,0,0,0);			//停止
[198]	LED(3);					//全 LED 点灯
[199]	Wait(1000);				//1 秒間待つ
[200]	}
[201]}
```

[プログラム解説]

・1 行目は最大に進めるミリ秒を定義します.

・2 行目はそれぞれの関数のプロトタイプ宣言をします.

・5～8 行目はそれぞれの変数の宣言をしています.

・9 行目は速さが高速で前進するミリ秒数を指定しています.

・12～13 行目は CPU の初期設定を行っています.

・15 行目は停止しています.

・16 行目は全ての LED が点灯します. .

・18 行目以降はメインループ内です.

・19～20 行目はカウント用変数を宣言しています.

・21 行目は結果を代入するための変数を宣言しています.

・22～23 行目は左センサと右センサの値をそれぞれ取得しています.

・25～26 行目は 0～6 までの 7 通りの数で分岐されているので, rand()関数で得られる数値も 0～7 の範囲に制限する必要があります. このような場合, 割り算の余りを求める「%」の演算子を使用します.

ここでは 7 個の変数がほしいので,

i=rand（）%7;

j=rand（）%7

という演算を行うと, rand（）の戻り値がどんな値であっても result は 0～7 の間の値

となります.

・29～93 行目はもし左センサが黒いラインを発見したら，変数 m に m+1 を代入し，ランダム関数を利用してランダムにロボットの回転角度が変更されます．0.5 秒高速で後退した後，緑色の LED が点灯し，それぞれの回転角度に応じて左旋回します.

・95～160 行目はもし右センサが黒いラインを発見したら，変数 j に j +1 を代入し，ランダム関数を利用してランダムにロボットの回転角度が変更されます．0.5 秒高速で後退した後，緑色の LED が点灯し，それぞれの回転角度に応じて右旋回します.

・161～165 行目はもし黒いラインが発見されなかったら高速の速さで前進し停止する関数を呼び出し，LED が全点灯します.

・176 行目は関数名と引数の説明をしています.

・51～76 行目は関数の宣言をしています．関数内は以下で説明します.

・178～184 行目はもし 9 行目で指定したミリ秒数より MAX_TIME の値のほうが大きかったら，モータが 9 行目で指定したミリ秒数だけ進み，オレンジ色のＬＥＤが点灯し，1 秒間停止するというものです.

・185～200 行目まではもし 9 行目で指定したミリ秒数より MAX_TIME の値のほうが小さかったら，変数 x に sec と MAX_TIME を割り算した結果，変数 y に sec と MAX_TIME を剰余した結果を代入します．これを求めることにより，割り算した結果だけ 188 行目の for 文内の処理を繰り返し，for 文が終了したら y ミリ秒作動します.

☆フローチャート☆

この問題では switch 文（29～94 行目）の部分のフローチャートを示しました.

switch 文のフローチャートの特性についても理解しましょう.

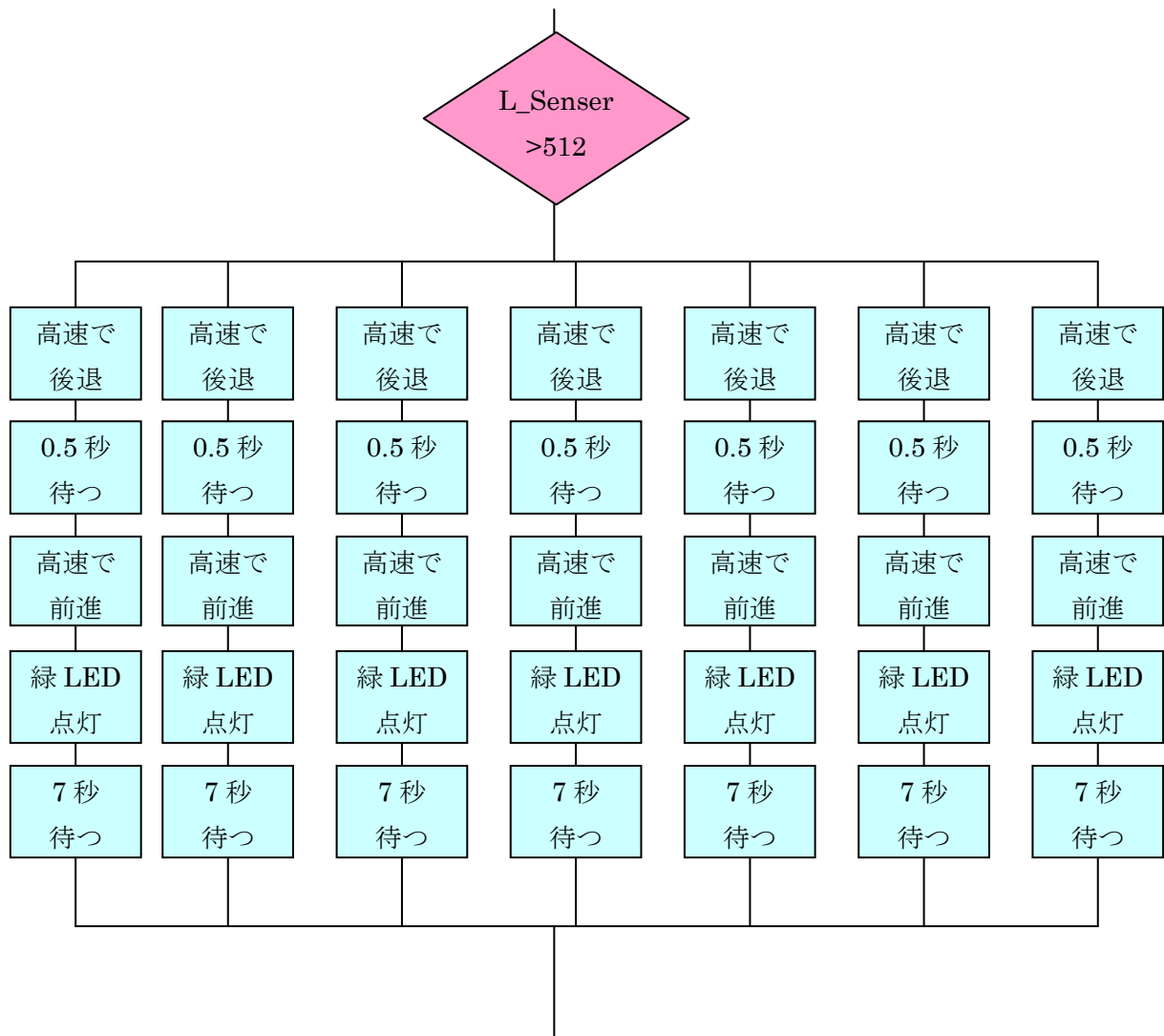



図 3-27

[参考資料]

H8 マイコンによる組込みプログラミング入門　ロボットで学ぶ C 言語

ヴイストン株式会社[編集] ロボット実習教材研究会[監修]